版权注意事项：
1、书籍版权归作者和出版社所有
2、本PDF仅限用于个人获取知识，进行私底下的知识交流
3、PDF获得者不得在互联网上以任何目的进行传播
4、如觉得书籍内容很赞，请购买正版实体书，支持作者
5、请于下载PDF后24小时内删除本PDF。

软件开发视频大讲堂
RUANJIANKAIFASHIPINDAJIANGTANG

八年锤炼
打造经典

DB2

从入门到精通

17小时语音视频讲解

明日科技 编著

☑实例资源库 ☑模块资源库 ☑项目资源库
☑面试资源库 ☑测试题库系统 ☑PPT电子课件



循序渐进，实战讲述

基础知识➪核心技术➪高级应用➪项目实战
204个学习实例，57个练习实例

海量资源，可查可练

除本书配套的17小时视频讲解外，根据学习顺序，光盘还额外配备如下海量开发资源库：

实例资源库(1010个实例)➪模块资源库(15个典型模块)➪项目资源库(15个项目案例)➪测试题库系统(596道测试题)➪面试资源库(369个面试真题)

在线解答，高效学习

QQ: 400 675 1066(可容纳10万人在线)

清华大学出版社

光盘“开发资源库”简明目录

☑ 实例资源库

☑ 模块资源库

- 构建开发环境
- 图文验证码模块
- 注册与登录验证模块
- 投票统计模块
- 上传下载模块
- 聊天室
- 搜索引擎模块
- RSS模块
- 网站留言簿

……

☑项目资源库

- 大学生就业求职网
- 电子商务系统
- 都市供求信息网
- 蜀玉网络购物商城
- 网络购物中心
- 博客网站
- 企业办公自动化
- 企业门户网站

……

☑ 面试资源库

- 职场人生
  - 第1部分　Java　程序员职业规划
  - 第2部分　Java　程序员面试技巧
  - 第3部分　Java　常见面试题
  - 第4部分　Java　企业面试真题汇编
    - 企业面试真题汇编（一）
    - 企业面试真题汇编（二）
    - 企业面试真题汇编（三）

……

☑ 测试题库系统

- Java 编程基础能力测试
- 数学及逻辑思维能力测试
  - 基本测试
  - 进阶测试
  - 高级测试
- 面试能力测试
  - 常规面试测试
- 编程英语能力测试

……

在线解答，高效学习

QQ: 400 675 1066（可容纳10万人在线）

官方网站: www.mingribook.com

软件开发视频大讲堂

DB2 从入门到精通

明日科技　编著

清华大学出版社

北　京

内 容 简 介

《DB2 从入门到精通》一书从初学者角度出发，通过通俗易懂的语言、丰富多彩的实例，详细介绍了使用 DB2 进行数据管理的各种技术。全书共分 3 篇 19 章，包括初识 DB2，DB2 的体系结构与 DB2 工具，实例和服务器管理，DB2 数据库基础，管理表空间和表，表数据的基本操作，数据库对象，DB2 数据查询技术，视图的使用，约束、索引和别名，SQL 过程语言，DB2 中的函数，存储过程和触发器，DB2 数据库安全处理，数据库的备份与恢复，PureXML 技术应用，XQuery 技术的使用，数据移动，事务与锁等内容。书中所有知识都结合具体实例进行介绍，涉及的程序代码给出了详细的注释，可以使读者轻松领会 DB2 管理数据库的精髓，快速提高数据库管理技能。

本书列举了大量的实例，所附 DVD 光盘内容有同步视频讲解、实例源程序、“实践与练习”答案等；本书的服务网站提供了模块库、案例库、题库、素材库、答疑服务。

本书内容详尽，实例丰富，非常适合作为 DB2 初学者的学习用书，也适合作为开发人员的查阅、参考资料。

本书封面贴有清华大学出版社防伪标签，无标签者不得销售。
版权所有，侵权必究。侵权举报电话：010-62782989　13701121933

图书在版编目（CIP）数据

DB2 从入门到精通/明日科技编著. —北京：清华大学出版社，2017
（软件开发视频大讲堂）
ISBN 978-7-302-45800-5

I. ①D…　II. ①明…　III. ①关系数据库系统　IV. ①TP311.138

中国版本图书馆 CIP 数据核字（2016）第 291072 号

责任编辑：赵洛育
封面设计：刘洪利
版式设计：魏　远
责任校对：王　云
责任印制：李红英

出版发行：清华大学出版社
　　网　　址：http://www.tup.com.cn，http://www.wqbook.com
　　地　　址：北京清华大学学研大厦 A 座　　**邮　　编**：100084
　　社 总 机：010-62770175　　**邮　　购**：010-62786544
　　投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
　　质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 装 者：清华大学印刷厂
经　　销：全国新华书店
开　　本：203mm×260mm　　**印　　张**：29　　**字　　数**：793 千字
　　（附海量开发资源库 DVD1 张）
版　　次：2017 年 6 月第 1 版　　**印　　次**：2017 年 6 月第1 次印刷
印　　数：1～5000
定　　价：79.80 元

产品编号：058899-01

前 言

Preface

丛书说明：“软件开发视频大讲堂”丛书（第1版）于2008年8月出版，因其编写细腻，易学实用，配备全程视频等特色，在软件开发类图书市场上产生了很大反响，绝大部分品种在全国软件开发零售图书排行榜中名列前茅，2009年多个品种被评为“全国优秀畅销书”。

“软件开发视频大讲堂”丛书（第2版）于2010年8月出版，出版后，绝大部分品种在全国软件开发类零售图书排行榜中依然名列前茅。丛书中多个品种被百余所高校计算机相关专业、软件学院选为教学参考书，在众多的软件开发类图书中成为最耀眼的品牌之一。丛书累计销售40多万册。

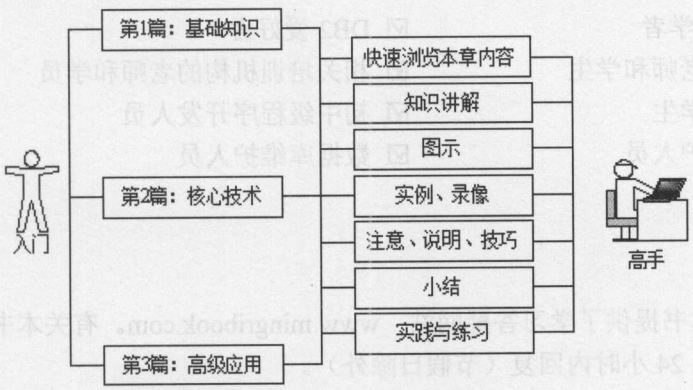
“软件开发视频大讲堂”丛书（第3版）于2012年8月出版，根据读者需要，增删了品种，重新录制了视频，提供了“入门学习→实例应用→模块开发→项目开发→能力测试→面试”等各个阶段的海量开发资源库。因丛书编写结构合理、实例选择经典实用，丛书迄今累计销售90多万册。

“软件开发视频大讲堂”丛书（第4版）在继承前3版所有优点的基础上，修正了前3版图书中发现的疏漏之处，并结合目前市场需要，进一步对丛书品种进行了完善，对相关内容进行了更新优化，使之更适合读者学习，为了方便教学，还提供了教学课件PPT。

DB2是IBM公司研制的一种关系型数据库系统，它主要应用于大型应用系统，具有较好的可伸缩性，可支持从大型机到单用户环境，可以应用于OS/2、Windows等平台。DB2提供了高层次的数据利用性、完整性、安全性、可恢复性，以及小规模到大规模应用程序的执行能力，具有与平台无关的基本功能和SQL命令。另外，它还采用了数据分级技术，能够使大型机数据很方便地下载到LAN数据库服务器，使得客户机/服务器用户和基于LAN的应用程序可以访问大型机数据，并使数据库本地以及远程连接透明化。

本书内容

本书提供了从DB2入门学员到编程高手所必备的各类知识，共分3篇，大体结构如下图所示。



第 1 篇：基础知识。本篇通过对初识 DB2，DB2 的体系结构与 DB2 工具，实例和服务器管理，DB2 数据库基础，管理表空间和表，表数据的基本操作，数据库对象等内容的介绍，并结合大量的图示、举例、录像等使读者快速掌握 DB2 的基础内容，为以后 DB2 数据库管理的学习奠定坚实的基础。

第 2 篇：核心技术。本篇介绍 DB2 数据查询技术，视图的使用，约束、索引和别名，SQL 过程语言，DB2 中的函数，存储过程和触发器等内容。学习完这一部分，读者能够对 DB2 数据库进行基本的日常管理和维护。

第 3 篇：高级应用。本篇介绍 DB2 数据库安全处理，数据库的备份与恢复，PureXML 技术应用，XQuery 技术的使用，数据移动，事务与锁等内容。学习完这一部分，能够实现对 DB2 数据库的安全处理，备份和恢复数据库、向 DB2 数据库导入数据等。

本书特点

- **由浅入深，循序渐进：**本书以初中级程序员为对象，先从 DB2 基础知识讲起，再学习 DB2 的核心技术，然后学习 DB2 的高级应用。讲解过程中步骤详尽，版式新颖，让读者在阅读中一目了然，从而快速掌握书中内容。
- **语音视频，讲解详尽：**书中提供了声图并茂的语音视频教学录像，读者可以在光盘中找到相应章节的录像。这些录像能够引导初学者快速入门，感受编程的快乐和成就感，增强进一步学习的信心，从而快速成为编程高手。
- **实例典型，轻松易学：**通过例子学习是最好的学习方式，本书通过“一个知识点、一个例子、一个结果”的模式，透彻详尽地讲述了实际开发中所需的各类知识。另外，为了便于读者阅读程序代码，快速学习编程技能，书中在关键代码处提供了注释。
- **精彩栏目，贴心提醒：**本书根据需要在各章使用了很多“注意”“说明”“技巧”等小栏目，让读者可以在学习过程中更轻松地理解相关知识点及概念，并轻松地掌握个别技术的应用技巧。
- **应用实践，随时练习：**书中几乎每章都提供了“实践与练习”环节，让读者能够通过对问题的解答重新回顾、熟悉所学的知识，举一反三，为进一步学习做好充分的准备。

读者对象

- ☑ 初学 DB2 的自学者
- ☑ DB2 爱好者
- ☑ 大中专院校的老师和学生
- ☑ 相关培训机构的老师和学员
- ☑ 做毕业设计的学生
- ☑ 初中级程序开发人员
- ☑ 程序测试及维护人员
- ☑ 数据库维护人员

读者服务

为了方便读者，本书提供了学习答疑网站：www.mingribook.com。有关本书的问题读者均可在网站上留言，我们力求在 24 小时内回复（节假日除外）。

致读者

本书由明日科技组织编写，主要编写人员有徐明明、李磊、王小科、王国辉、申小琦、董刚、赛奎春、房德山、杨丽、高春艳、辛洪郁、周佳星、张鑫、张宝华、葛忠月、刘杰、白宏健、张雳霆、马新新、冯春龙、宋万勇、李文欣、王东东、柳琳、王盛鑫、杨柳、赵宁、王佳雪、于国良、李彦骏、王泽奇、贾景波、谭慧、李丹、吕玉翠、孙巧辰、赵颖、江玉贞、周艳梅、房雪坤、裴莹、郭铁、张金辉、王敬杰、高茹、李贺、陈威、高飞、刘志铭、高润岭、于国槐、郭锐、郭鑫、邹淑芳、李根福、杨贵发、王喜平等，在编写本书的过程中，我们以科学、严谨的态度，力求精益求精，但错误、疏漏之处在所难免，敬请广大读者批评指正。我们的服务邮箱是 tmoonbook@sina.com，th_press@263.net。读者在阅读本书时，如果发现错误或遇到问题，可以发送电子邮件及时与我们联系，我们会尽快给予答复。

感谢您购买本书，希望本书能成为您编程路上的领航者。

“零门槛”编程，一切皆有可能。祝读书快乐！

编　者

目 录

Contents

第1篇 基 础 知 识

第 1 章 初识 DB2 2

视频讲解：31 分钟

1.1 DB2 的发展历史 3

1.2 关系型数据库的基本理论 4

1.2.1 数据模型的概念 4

1.2.2 常见的数据模型 4

1.2.3 关系型数据库与数据库管理系统 5

1.2.4 关系型数据库的 E-R 模型 6

1.2.5 关系型数据库的设计范式 8

1.2.6 关系数据库的设计原则 10

1.3 其他常见关系数据库 10

1.3.1 Access 数据库 11

1.3.2 SQL Server 数据库 11

1.3.3 Oracle 数据库 11

1.4 DB2 的安装与配置 11

1.4.1 DB2 的安装 11

1.4.2 验证 DB2 的安装 19

1.5 小结 21

1.6 实践与练习 21

第 2 章 DB2 的体系结构与 DB2 工具 22

视频讲解：66 分钟

2.1 DB2 体系结构 23

2.2 对象层次关系 24

2.3 数据访问过程 24

2.4 DB2 命令工具的使用 25

2.4.1 命令编辑器 25

2.4.2 命令窗口 26

2.4.3 命令行处理器 26

2.4.4 “命令编辑器”“命令窗口”“命令行处理器”用法的区别 27

2.5 DB2 常用工具 28

2.5.1 配置助手 28

2.5.2 控制中心 29

2.5.3 工具设置 31

2.5.4 向导 32

2.5.5 信息中心 32

2.5.6 运行状况中心 33

2.5.7 任务中心 33

2.5.8 复制中心 34

2.5.9 日志 35

2.6 小结 36

2.7 实践与练习 36

第 3 章 实例和服务器管理 37

视频讲解：26 分钟

3.1 实例 38

3.1.1 什么是 DB2 实例 38

3.1.2 创建实例 38

3.1.3 实例目录 39

3.1.4 列出实例 40

3.1.5 启动/停止实例 40

3.1.6 更新实例 40

3.1.7 删除实例 41

3.1.8 配置实例 41

3.2 管理服务器 42

3.2.1 管理服务器的概念 42

3.2.2 创建管理服务器 42

3.2.3 管理服务器的相关命令 43
3.2.4 删除管理服务器 44
3.2.5 配置管理服务器 44
3.3 小结 45
3.4 实践与练习 45

第 4 章 DB2 数据库基础 46
视频讲解：30 分钟
4.1 数据库结构 47
4.2 数据库基本概念 48
4.3 数据库的创建、编目和删除 49
4.3.1 创建数据库 49
4.3.2 SQL 语句中的注释符 54
4.3.3 编目数据库 54
4.3.4 删除数据库 64
4.4 小结 65
4.5 实践与练习 65

第 5 章 管理表空间和表 66
视频讲解：53 分钟
5.1 表空间的创建、修改和删除 67
5.1.1 创建表空间 68
5.1.2 修改表空间 70
5.1.3 删除表空间 72
5.2 表的创建、修改和删除 73
5.2.1 DB2 数据类型 74
5.2.2 创建表 77
5.2.3 修改表 84
5.2.4 删除表 91
5.3 小结 92
5.4 实践与练习 92

第 6 章 表数据的基本操作 93
视频讲解：30 分钟
6.1 SQL 语言的概述 94
6.1.1 SQL 语言的特点 94
6.1.2 SQL 语言的分类 95
6.1.3 SQL 语言的编写规则 95
6.2 使用界面操作表数据 96
6.3 插入记录 98
6.3.1 用界面方式插入记录 98
6.3.2 用命令方式插入记录 99
6.4 修改记录 103
6.4.1 用界面方式修改记录 103
6.4.2 用命令方式修改记录 104
6.5 删除记录 107
6.5.1 用界面方式删除记录 108
6.5.2 用命令方式删除记录 108
6.6 小结 109
6.7 实践与练习 109

第 7 章 数据库对象 110
视频讲解：47 分钟
7.1 数据库对象概述 111
7.2 模式 111
7.2.1 使用控制中心的方式创建模式 112
7.2.2 使用命令的方式创建模式 114
7.3 表 115
7.3.1 表的维护 115
7.3.2 表约束 116
7.3.3 表状态 118
7.3.4 表压缩 119
7.3.5 表分区 120
7.4 索引 121
7.4.1 B+树结构 121
7.4.2 索引的好处 122
7.4.3 合适的索引 124
7.4.4 复合索引 124
7.5 其他常用数据库对象 126
7.5.1 视图 127
7.5.2 昵称 127
7.5.3 序列 127
7.5.4 自增字段 129
7.6 小结 131
7.7 实践与练习 132

第2篇 核心技术

第8章 DB2数据查询技术 134
视频讲解：78分钟
8.1 选择、投影、连接和集合运算 135
8.1.1 选择 135
8.1.2 投影 136
8.1.3 连接 136
8.1.4 集合运算 137
8.2 检索数据 139
8.2.1 简单查询 140
8.2.2 筛选查询 145
8.2.3 分组查询 150
8.2.4 排序查询 154
8.2.5 多表关联查询 156
8.2.6 使用聚集函数计算列值 162
8.3 子查询的用法 164
8.3.1 子查询 164
8.3.2 单行子查询 166
8.3.3 多行子查询 166
8.3.4 关联子查询 168
8.4 小结 169
8.5 实践与练习 169

第9章 视图的使用 170
视频讲解：37分钟
9.1 视图概述 171
9.1.1 视图与表的关系 171
9.1.2 视图的优点 172
9.1.3 使用视图的注意事项 172
9.2 创建视图 172
9.2.1 使用界面方式创建视图 172
9.2.2 使用命令方式创建视图 177
9.2.3 创建特殊类型视图 178
9.3 查询视图 185
9.4 更新视图 188
9.4.1 使用界面方式操作视图数据 188
9.4.2 使用命令方式操作视图数据 191
9.5 修改视图的注释 193
9.5.1 使用界面方式为视图CJB_VIEW1添加注释 193
9.5.2 使用命令方式为视图添加注释 194
9.6 删除视图 194
9.6.1 使用界面方式删除视图 194
9.6.2 使用命令方式删除视图 195
9.7 视图的维护 196
9.8 小结 196
9.9 实践与练习 196

第10章 约束、索引和别名 197
视频讲解：46分钟
10.1 默认值和约束 198
10.1.1 默认值 198
10.1.2 约束 199
10.2 索引 205
10.2.1 索引的概念 205
10.2.2 索引的分类 206
10.2.3 创建索引 206
10.2.4 查看已创建的索引 212
10.2.5 修改索引 214
10.2.6 重命名索引 214
10.2.7 删除索引 215
10.3 别名 216
10.3.1 创建别名 217
10.3.2 使用别名 218
10.3.3 删除别名 218
10.4 小结 218
10.5 实践与练习 219

第11章 SQL过程语言 220
视频讲解：120分钟
11.1 SQL PL语言 221

11.1.1 SQL PL 语言元素 221
11.1.2 VALUES 语句 222
11.2 数据类型 223
11.2.1 系统数据类型 223
11.2.2 创建单值数据类型 223
11.2.3 创建结构数据类型 227
11.2.4 创建数组数据类型 230
11.3 声明变量 230
11.4 赋值 232
11.4.1 赋值语句的语法 232
11.4.2 专用寄存器 232
11.5 游标 233
11.5.1 游标的概念 234
11.5.2 声明游标 234
11.5.3 打开游标 236
11.5.4 读取游标 237
11.5.5 关闭游标 237
11.5.6 游标和结果集 238
11.6 流程控制语句 239
11.6.1 条件语句 239
11.6.2 迭代语句与退出循环语句 243
11.6.3 RETURN 语句 248
11.7 异常处理机制 248
11.7.1 声明有名称的条件 248
11.7.2 声明条件处理程序 249
11.7.3 强制发出异常 249
11.8 MERGE 语句 250
11.9 小结 253
11.10 实践与练习 253

第 12 章 DB2 中的函数 254
视频讲解：38 分钟
12.1 系统内置函数 255
12.1.1 系统内置函数介绍 255
12.1.2 常用系统内置函数 256
12.2 用户定义函数 261
12.2.1 创建和调用用户定义函数 261
12.2.2 用户定义函数的删除 267
12.3 小结 267
12.4 实践与练习 267

第 13 章 存储过程和触发器 268
视频讲解：71 分钟
13.1 存储过程 269
13.1.1 存储过程的概念 269
13.1.2 存储过程的类型 270
13.1.3 SQL 存储过程的创建与执行 270
13.1.4 存储过程的重载 284
13.1.5 存储过程的查询、修改和删除 285
13.2 触发器 286
13.2.1 触发器的概念 287
13.2.2 触发器的类型 287
13.2.3 用 CREATE TRIGGER 语句创建触发器 288
13.2.4 查询、修改和删除触发器 296
13.3 小结 300
13.4 实践与练习 300

第 3 篇 高级应用

第 14 章 DB2 数据库安全处理 302
视频讲解：72 分钟
14.1 DB2 数据库安全概述 303
14.2 认证机制 303
14.3 身份验证和授权 304
14.4 DB2 身份验证 305
14.4.1 DB2 身份验证方式 305
14.4.2 其他身份验证参数 306
14.4.3 在客户机-服务器环境中设置身份验证 307
14.5 管理权限 314
14.5.1 管理权限级别 314

14.5.2 授予或撤销实例级权限 316
14.5.3 授予或撤销数据库级权限 319
14.6 特权 322
14.6.1 特权级别 322
14.6.2 授予和撤销特权 323
14.6.3 隐式特权 328
14.7 LBAC 凭证 329
14.8 GET AUTHORIZATIONS 命令 330
14.9 使用模式控制对数据库对象的访问 331
14.10 小结 331
14.11 实践与练习 331

第 15 章 数据库的备份与恢复 332
视频讲解：53 分钟
15.1 备份与恢复 333
15.2 恢复操作的种类 333
15.3 事务日志记录 334
15.3.1 为什么需要日志 334
15.3.2 事务日志记录的概念 335
15.3.3 主日志文件和辅助日志文件 335
15.3.4 循环日志记录和归档日志记录 336
15.3.5 修改事务日志模式 337
15.4 备份 338
15.4.1 离线备份 339
15.4.2 在线备份 339
15.4.3 表空间备份 340
15.4.4 增量备份 341
15.5 恢复 342
15.5.1 版本恢复 342
15.5.2 前滚恢复 347
15.5.3 崩溃恢复 348
15.6 常见的恢复场景 349
15.6.1 整个数据库意外删除和损坏 349
15.6.2 表空间容器意外丢弃或损坏 353
15.6.3 恢复到时间点 356
15.7 小结 358
15.8 实践与练习 358

第 16 章 PureXML 技术应用 359
视频讲解：55 分钟
16.1 pureXML 技术简介 360
16.2 创建可以存储 XML 数据的 DB2 数据库 361
16.3 创建具有 XML 列的表 361
16.4 样本数据 362
16.5 XML 模式存储库 366
16.5.1 XML 模式存储库概念 366
16.5.2 增大应用程序堆栈大小配置参数 366
16.5.3 增大代理程序堆栈大小配置参数 367
16.5.4 编写 XML 模式文档 369
16.5.5 向 XSR 注册 XML 模式 370
16.5.6 删除 XSR 中的 XML 模式 376
16.6 将 XML 文档插入到 XML 类型的列中 378
16.6.1 插入时针对 XML 模式验证 XML 文档 378
16.6.2 插入时不验证 XML 文档 379
16.7 小结 379
16.8 实践与练习 380

第 17 章 XQuery 技术的使用 381
视频讲解：49 分钟
17.1 XQuery 基础 382
17.1.1 XQuery 查询的组成部分 382
17.1.2 基本规定 383
17.1.3 XML 名称空间和 QName 383
17.1.4 序言 384
17.1.5 表达式 386
17.2 查询 XML 数据 394
17.2.1 界面查询表中 XML 数据 394
17.2.2 如何选择查询 XML 数据的语言 395
17.2.3 用普通 SQL 查询 XML 数据 396
17.2.4 XQuery 和嵌入了 SQL 的 XQuery 398
17.2.5 SQL/XML 399
17.3 更新 XML 数据 402
17.3.1 使用命令更新 XML 数据 402
17.3.2 变换表达式 403

17.4 小结 405
17.5 实践与练习 405

第 18 章 数据移动 406
视频讲解：68 分钟
18.1 数据移动概述 407
18.2 文件格式 407
18.2.1 DEL 格式 408
18.2.2 ASC 格式 408
18.2.3 PC/IXF 格式 408
18.2.4 WSF 格式 408
18.2.5 CURSOR 格式 409
18.3 数据的导出 409
18.3.1 使用 EXPORT 导出数据 409
18.3.2 EXPORT 命令格式及选项 413
18.4 数据的导入 416
18.4.1 使用 IMPORT 导入数据 416
18.4.2 IMPORT 命令格式及选项 420
18.5 数据的装入 422
18.5.1 装入操作的 4 个阶段 422
18.5.2 使用 LOAD 清空表数据 423
18.5.3 LOAD 实用程序 424
18.6 复制表 429
18.7 常见问题 431
18.7.1 加载的数据是 Excel 格式 431
18.7.2 导出/加载的数据不是逗号/双引号分隔 432
18.7.3 文件中的列比要导入的表中的字段多 432
18.7.4 导入/导出大字段（LOB） 432
18.7.5 导入 identity 数据 433
18.7.6 数据移动出现乱码 433
18.7.7 表数据从一个表空间迁移到另一个表空间 434
18.8 小结 435
18.9 实践与练习 435

第 19 章 事务与锁 436
视频讲解：27 分钟
19.1 数据库事务 437
19.2 事务日志记录 437
19.2.1 控制事务日志记录的参数 438
19.2.2 日志文件存储的位置 441
19.3 并发性控制 442
19.4 锁 443
19.4.1 锁的基本概念 443
19.4.2 行级锁和表级锁的模式 444
19.4.3 获取不同级别的锁的方法 446
19.5 隔离级别 447
19.5.1 隔离级别的种类 447
19.5.2 设置数据库的事务隔离级别 448
19.5.3 重写隔离级别（WITH 从句） 448
19.5.4 继承隔离级别 449
19.5.5 锁避免和隔离 449
19.6 小结 449
19.7 实践与练习 450



基础知识

- 第 1 章　初识 DB2
- 第 2 章　DB2 的体系结构与 DB2 工具
- 第 3 章　实例和服务器管理
- 第 4 章　DB2 数据库基础
- 第 5 章　管理表空间和表
- 第 6 章　表数据的基本操作
- 第 7 章　数据库对象

本篇通过对初识 DB2、DB2 的体系结构与 DB2 工具、实例和服务器管理、DB2 数据库基础、管理表空间和表、表数据的基本操作、数据库对象等内容的介绍，并结合大量的图示、举例、录像等使读者快速掌握 DB2 的基础内容，为以后 DB2 数据库管理的学习奠定坚实的基础。

第 1 章

初识 DB2

（视频讲解：31 分钟）

本章主要介绍数据库的相关概念，包括 DB2 数据库的发展史概述、数据模型的概念、关系型数据库的 E-R 模型、关系型数据库的设计范式、关系数据库的设计原则、其他常见数据库以及 DB2 的安装与配置等内容。通过本章的学习，读者应该掌握数据库系统、数据模型以及数据库设计范式等概念，对比常见的关系数据库。

通过阅读本章，您可以：

- 了解 DB2 数据库的发展
- 掌握关系型数据库的基本理论
- 了解其他常见的关系数据库
- 掌握 DB2 数据库的安装与配置

1.1　DB2 的发展历史

对于每个最终站在领奖台上泪水盈面的奥运冠军来说，为此刻他（她）也许已经付出了 5 年甚至 10 年的艰苦努力。相对这些人类的冠军们，这个世界还有另外一种意义上的冠军，他们虽没有泪水，却依然在历史上留下了非凡的轨迹——DB2 就是这类冠军中的一员，这个数据库领域里当之无愧的冠军，它已用了足足 25 年来描绘自己的足迹。

DB2 LUW（IBM DB2 Database for Linux、UNIX and Windows，在本书后续章节中统一简写为 DB2）是 IBM 公司研制的一种关系型数据库系统。DB2 主要应用于大型应用系统，具有较好的可伸缩性。DB2 提供了高层次的数据利用性、完整性、安全性、可恢复性，以及小规模到大规模应用程序的执行能力，具有与平台无关的基本功能和 SQL 命令运行环境。

1968 年，在 IBM 360 计算机上研制成功了 IMS V1，这是第一个也是最著名的和最为典型的层次型数据库管理系统。

1970 年，IBM 公司的研究员 E.F.Codd 发表了业界第一篇关于关系型数据库理论的论文《A Relational Model of Data for Large Shared Data Banks》，首次提出了关系模型的概念，这篇论文是计算机科学史上最重要的论文之一，奠定了 Codd 博士“关系数据库之父”的地位。

1973 年，IBM 研究中心启动 System R 项目，为 DB2 的诞生打下了良好基础。System R 是 IBM 研究部门开发的一种产品，这种原型语言促进了技术的发展并将 DB2 带到了商业市场。

1981 年，IBM 的研究员 E.F.Codd 发明的关系型数据库模型获得了 ACM 图灵奖，这是计算机科学界的最高荣誉。Codd 博士是继查尔斯·巴赫曼（Charles W. Bachman）之后，又一位由于在数据库领域做出巨大贡献而获此殊荣的计算机科学家。

1983 年，IBM 推出的第一款面向大型企业的商业化关系数据库管理系统（Database2 for MVS），内部代号为“Eagle”，于是 DB2 正式诞生。

1986 年，IBM 发布了 System/38 V7，该系统首次配置查询优化器，能对应用的存取计划进行优化，而后 DB2 又完成了到 OS/2 V1.0 的扩展，这是 IBM 第一次把关系型数据库处理能力扩展到微机系统。也是 DB2 for OS/2、UNIX and Windows 的雏形。

1995 年，IBM 发布了 DB2 Common Server V2，这是第一个能够在多个平台上运行的“对象-关系型数据库”（ORDB）产品，并能够对 Web 提供充分支持。DataJoiner for AIX 也诞生在这一年，该产品赋予了 DB2 对异构数据库的支持能力。

1996 年，IBM 发布 DB2 V2.1.2，这是第一个真正支持 Java 和 JDBC 的数据库产品。

1997 年，IBM 发布了可以支持 Web 的 DB2 for OS/390 V5，这是当时唯一能够支持 64000 个并发用户和百 TB 级别的数据库产品。

1999 年，IBM 为了对移动计算提供支持，发布了 DB2 UDB 卫星版和 DB2 Everywhere。

2000 年，IBM 发布了 DB2 XML Extender，成为在业界第一个为数据库提供内置 XML 支持的厂商。同年 IBM 将 Visual Warehouse 集成到 DB2 中，为 DB2 提供了内置的数据仓库管理功能。

2001 年，IBM 以 10 亿美金收购了 Informix 的数据库业务，这次收购扩大了 IBM 的分布式数据库业务。

2002 年，IBM 宣布计划收购 Rational Software Corp，从而使得 IBM 软件能够支持从设计、开发、部署到管理和维护的完整过程。

DB2 的诞生不仅促进了与关系数据库概念相关的数学和科学的发展，还创造性地开发出一种极具影响力的全新软件类型。今天 DB2 已经发展成为 IBM 信息管理（IM）软件组合的重要组成部分。在 IBM 信息随需应变策略和体系结构中，DB2 扮演数据基础服务平台的重要角色，并且已经发展成同时支持传统关系数据和 XML 的混合型数据服务器，DB2 数据库产品及解决方案已经广泛应用在金融、电信、制造、零售等行业。

1.2 关系型数据库的基本理论

数据库技术是应对信息资源（即大量数据）的管理需求而产生的，随着信息技术的不断发展，尤其是人类迈入网络时代后，社会信息资源在爆炸式地增长，对数据管理技术也随之不断地提出更高的要求。数据库管理技术先后经历了人工管理、文件系统、数据库系统 3 个阶段。在数据库系统中，数据模型主要有层次模型、网状模型和关系模型 3 种（另外一种面向对象模型还处在探索研究中），目前理论成熟、使用普及的模型就是关系模型——关系型数据库的理论基础。

1.2.1 数据模型的概念

数据模型是数据库系统的核心与基础，是关于描述数据与数据之间的联系、数据的语义、数据一致性约束的概念性工具的集合。

数据模型通常是由数据结构、数据操作和完整性约束 3 部分组成的。

- ☑ 数据结构：是对系统静态特征的描述。描述对象包括数据的类型、内容、性质和数据之间的相互关系。
- ☑ 数据操作：是对系统动态特征的描述。是对数据库中各种对象实例的操作。
- ☑ 完整性约束：是完整性规则的集合。它定义了给定数据模型中数据及其联系所具有的制约和依存规则。

1.2.2 常见的数据模型

常用的数据库数据模型主要有网状模型、层次模型和关系模型，下面分别进行介绍。

（1）网状模型：用有向图结构表示实体类型及实体间联系的数据模型称为网状模型。用网状模型编写应用程序极其复杂，数据的独立性较差，如图 1.1 所示。

（2）层次模型：用树形结构表示实体类型及实体间联系的数据模型称为层次模型，它具有以下特点，如图 1.2 所示。

- ☑ 每棵树有且仅有一个无双亲节点，称为根。
- ☑ 树中除根外所有节点有且仅有一个双亲。

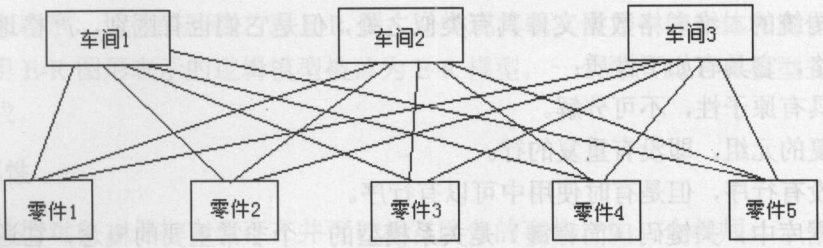


图 1.1　按网状模型组织的数据示例

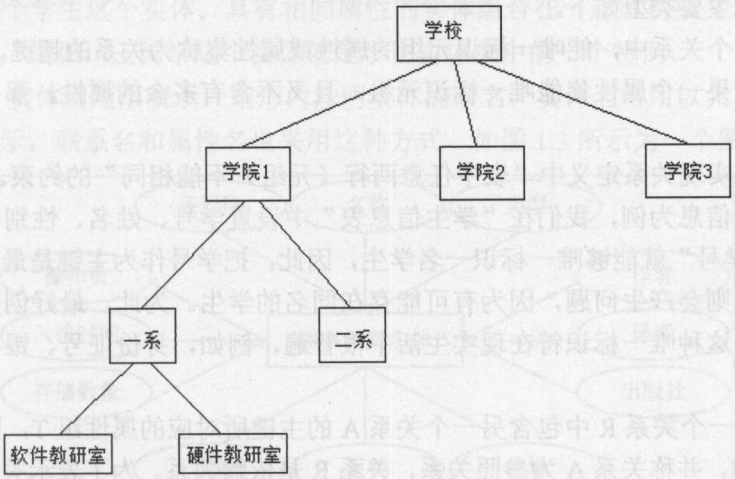


图 1.2　按层次模型组织的数据实例

（3）关系模型：以二维表来描述数据。关系模型中，每个表有多个字段列和记录行，每个字段列有固定的属性（数字、字符、日期等）。关系模型数据结构简单、清晰、具有很高的数据独立性，因此是目前主流的数据库数据模型。

关系中的数据约束如下。

☑ 实体完整性约束：约束关系的主键中属性值不能为空值。

☑ 参照完整性约束：关系之间的基本约束。

☑ 用户定义的完整性约束：它反映了具体应用中数据的语义要求。

1.2.3　关系型数据库与数据库管理系统

关系型数据库是建立在关系模型基础上的数据库，借助于集合代数等数学概念和方法来处理数据库中的数据，现实世界中的各种实体以及实体之间的各种联系均可用关系模型来表示。

在关系数据模型中，关系可以看成由行和列交叉组成的二维表格，表中一行称为一个元组，可以用来标识实体集（实体的集合）中的一个实体。表中的列称为属性，给每一列起一个名称即为属性名，表中的属性名不能相同。列的取值范围称为域，同列具有相同的域，不同的列也有相同的域。表中任意两行（元组）不能相同。能唯一标识表中不同行的属性或属性组（即多个属性的组合）称为主键或复合主键。

尽管关系与传统的二维表格数据文件具有类似之处，但是它们也有区别，严格地说，关系是一种规范化的二维表格，它具有如下性质：

- ☑ 属性值具有原子性，不可分解。
- ☑ 没有重复的元组，即没有重复的行。
- ☑ 理论上没有行序，但是有时使用中可以有行序。

在关系型数据库中，关键码（简称键）是关系模型的一个非常重要的概念，它通常是行（元组）的一个或几个列（属性）。如果键是由一个列组成，则称之为唯一键；若是由多个列（属性）组成，则称之为复合键，键的主要类型如下。

- ☑ 超键：在一个关系中，能唯一标识元组的属性或属性集称为关系的超键。
- ☑ 候选键：如果一个属性集能唯一标识元组，且又不含有多余的属性，那么这个属性集称为关系的候选键。
- ☑ 用主键可以实现关系定义中“表中任意两行（元组）不能相同”的约束。

这里以管理学生信息为例，我们在“学生信息表”中设置学号、姓名、性别、年龄、院系、班级等列。在该表中，“学号”就能够唯一标识一名学生，因此，把学号作为主键是最佳的选择，而如果把“姓名”列作为主键则会产生问题，因为有可能存在同名的学生。为此，最好创建一个单独的键将其明确地指定为主键，这种唯一标识符在现实生活中很普遍，例如，身份证号、银行卡号、手机号、发票号等。

- ☑ 外键：如果一个关系 R 中包含另一个关系 A 的主键所对应的属性组 T，则称此属性组 T 为关系 R 的外键，并称关系 A 为参照关系，关系 R 是依赖关系。为了表示关联，可以将一个关系的主键作为属性放入另外一个关系中，第二个关系中的那些属性就称为外键。

这里以商品销售为例，在填写一张商品销售单时，可以将商品销售信息分为两大类：第一类是单据的主体信息（销售主表），例如，销售单号、销售金额、销售日期、收款人；第二类是单据的明细信息（销售明细表），例如，商品序号、商品名称、商品数量等。在数据库的“销售主表”中通常以“销售单号”作为主键；在“销售明细表”中，为了标识被销售出去的商品隶属于哪张单据，需要对每一条商品销售记录标明“单据编号”。在这种情况下，销售明细表中的“销售单号”就被称为外键，因为“销售单号”是其所在表以外（主体表）的一个主键。

当出现外键时，主键与外键的列名称可以是不同的。但必须要求它们的值集相同，即“销售明细表”中出现的“销售单号”一定要和主体表中的值匹配。

对于上面提到的“二维表格”中存储的数据信息，通常以物理文件的形式存储在磁盘上，这种物理文件称之为“数据文件”，用户会使用一种数据库软件实现与磁盘上的数据文件进行交互，这种数据库软件就被称之为数据库管理系统（英文缩写为 DBMS）。DBMS 是建立在操作系统基础上的，它可以实现对数据库文件进行统一管理和控制。用户对数据库提出的访问请求都是由 DBMS 来处理的。另外，DBMS 还提供了多种用于管理数据的实用工具。

1.2.4 关系型数据库的 E-R 模型

在设计关系型数据库时，首先需要为它建立逻辑模型。关系型数据库的逻辑模型可以通过实体和

关系组成的图形来表示，这种图形称之为 E-R 图，它实现将现实世界中的实体和实体之间的联系转换为逻辑模型。使用 E-R 图形表示的逻辑模型被称为 E-R 模型，一个标准的 E-R 模型主要由实体、属性和联系 3 部分组成。

1. 实体和属性

实体是一个数据对象，是指客观存在并可以相互区分的事物，如一个教师、一个学生、一个雇员等。每个实体由一组属性来表示，如一个具体的学生拥有学号、姓名、性别和班级等属性，其中学号可以唯一标识具体某个学生这个实体。具有相同属性的实体组合在一起就构成实体集，而实体则是实体集中的某一个特例，例如，王同学这个实体就是学生实体集中的一个特例。

在 E-R 模型中，实体用矩形表示，矩形内注明实体的命名。实体名常用以大写字母开头的有具体意义的英文名词来表示，联系名和属性名也采用这种方式。如图 1.3 所示为一个图书档案的 E-R 图。

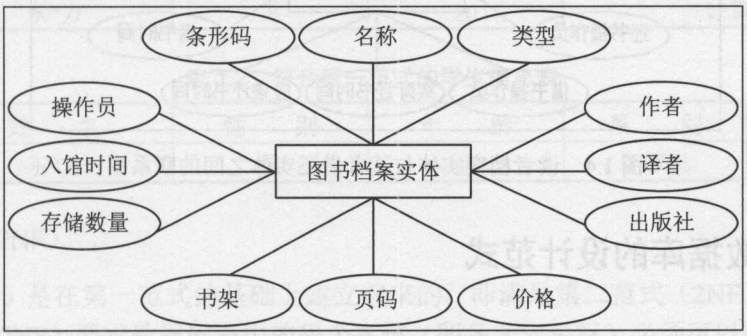


图 1.3　图书档案实体 E-R 图

2. 联系

在实际应用中，实体之间是存在联系的，这种联系必须在逻辑模型中表现出来。在 E-R 模型中，联系用菱形表示，菱形框内写明“联系名”，并用“连接线”将有关实体连接起来，同时在“连接线”的旁边标注上联系的类型，两个实体之间的联系类型可以分为以下 3 类。

- ☑ 一对一：若对于实体集 A 中的每一个实体，在实体集 B 中最多有一个实体与之相关，反之亦然，则称实体集 A 与实体集 B 具有一对一的联系，可标记联系为 1:1。
- ☑ 一对多：若对于实体集 A 中的每一个实体，在实体集 B 中有多个实体与之相关；反之，对于实体集 B 中的每一个实体，实体集 A 中最多有一个实体与之相关，则称实体集 A 与实体集 B 具有一对多的联系，可标记联系为 1:n。
- ☑ 多对多：若对于实体集 A 中的每一个实体，在实体集 B 中有多个实体与之相关；反之，对于实体集 B 中的每一个实体，实体集 A 中也有多个实体与之相关，则称实体集 A 与实体集 B 具有多对多的联系，可标记联系为 m:n。

例如，一个读者可以有多个图书借还记录，而一个借还记录只能隶属于一个读者，这样“读者档案实体”与“读者借还实体”之间就存在一对多的联系（即 1:n），那么这两个实体之间的联系如图 1.4 所示。

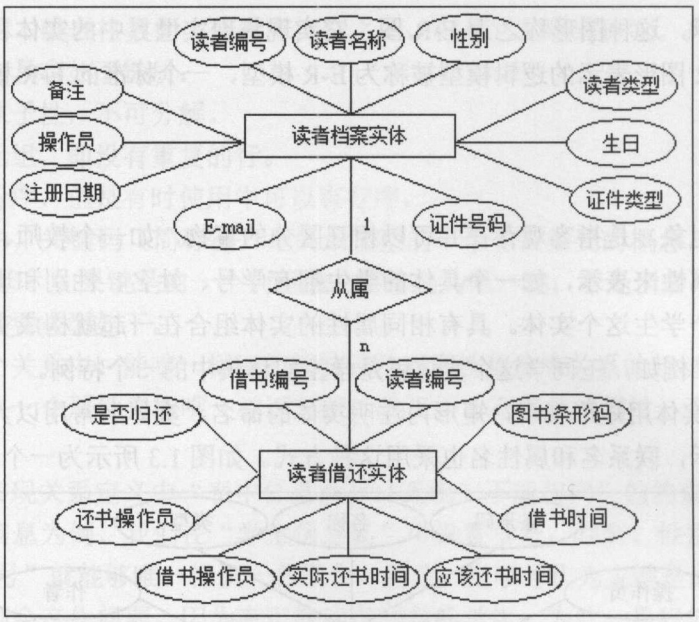


图 1.4　读者档案实体与读者借还实体之间的联系图

1.2.5　关系型数据库的设计范式

在数据库中，数据之间存在着密切的联系。关系型数据库由相互联系的一组关系所组成，每个关系包括关系模式和关系值两个方面。关系模式是对关系的抽象定义，给出关系的具体结构；关系的值是关系的具体内容，反映关系在某一时刻的状态。一个关系包含许多元组（记录行），每个元组都是符合关系模式结构的一个具体值，并且都分属于相应的属性。在关系数据库中的每个关系都需要进行规范化，使之达到一定的规范化程度，从而提高数据的结构化、共享性、一致性和可操作性。

规范化是把数据库组织成在保持存储数据完整性的同时最小化冗余数据的结构的过程。规范化的数据库必须符合关系模型的范式规则。范式可以防止在使用数据库时出现不一致的数据，并防止数据丢失。关系模型的范式有第一范式（1NF）、第二范式（2NF）、第三范式（3NF）、第四范式（4NF）、第五范式（5NF）、第六范式（6NF）和 BCNF 范式等多种。通常数据库只要满足前 3 个范式就足够用了，下面举例介绍前 3 个范式。

1. 第一范式（1NF）

第一范式是第二和第三范式的基础，是最基本的范式。第一范式包括下列指导原则：

☑ 数据组的每个属性只可以包含一个值。

☑ 关系中的每个数组必须包含相同数量的值。

☑ 关系中的每个数组一定不能相同。

在任何一个关系数据库中，第一范式是对关系模式的基本要求，不满足第一范式（1NF）的数据库就不是关系型数据库。

如果数据表中的每一个列都是不可再分割的基本数据项——即同一列中不能有多个值，那么就称

此数据表符合第一范式，由此可见第一范式具有不可再分解的原子特性。

在第一范式中，数据表的每一个行只包含一个实体的信息，并且每一行的每一列只能存放实体的一个属性。例如，对于学生信息，不可以将学生实体的所有属性信息（如学号、姓名、性别、年龄、班级等）都放在一个列中显示，也不能将学生实体的两个或多个属性信息放在一个列中显示，学生实体的每个属性信息都放在一个列中显示。

如果数据表中的列信息都符合第一范式，那么在数据表中的字段都是单一的，不可再分的。如表 1.1 就是不符合第一范式的学生信息表，因为“班级”列中包含了“系别”和“班级”两个属性信息，这样“班级”列中的信息就不是单一的，是可以再分的；而表 1.2 就是符合第一范式的学生信息表，它将原“班级”列的信息拆分到“系别”列和“班级”列中。

表 1.1　不符合第一范式的学生信息表

学　号	姓　名	性　别	年　龄	班　级
9527	东*方	男	20	计算机系 3 班

表 1.2　符合第一范式的学生信息表

学　号	姓　名	性　别	年　龄	系　别	班　级
9527	东*方	男	20	计算机	3 班

2. 第二范式（2NF）

第二范式（2NF）是在第一范式的基础上建立起来的，即满足第二范式（2NF）必先满足第一范式（1NF）。第二范式（2NF）要求数据库表中的每个实体（即各个记录行）必须可以被唯一地区分。为实现区分各行记录通常需要为表设置一个“区分列”，用以存储各个实体的唯一标识。在学生信息表中，设置了“学号”列，由于每个学生的编号都是唯一的，因此每个学生可以被唯一地区分（即使学生存在重名的情况），那么这个唯一属性列被称为主关键字或主键。

第二范式要求实体的属性完全依赖于主关键字，即不能存在仅依赖主关键字一部分的属性，如果存在，那么这个属性和主关键字的这一部分应该分离出来形成一个新的实体，新实体与原实体之间是一对多的关系。

例如，这里以“员工工资信息表”为例，若以（员工编码、岗位）为组合关键字（即复合主键），就会存在如下决定关系。

（员工编码，岗位）→（决定）（姓名、年龄、学历、基本工资、绩效工资、奖金）

在上面的决定关系中，还可以进一步拆分为如下两种决定关系。

（员工编码）→（决定）（姓名、年龄、学历）
（岗位）→（决定）（基本工资）

其中，员工编码决定了员工的基本信息（包括姓名、年龄、学历等），而岗位决定了基本工资，所以这个关系表不满足第二范式。

对于上面的这种关系，可以把上述两个关系表更改为如下 3 个表。

☑ 员工档案表：EMPLOYEE（员工编码，姓名，年龄，学历）

☑ 岗位工资表：QUARTERS（岗位，基本工资）

☑ 员工工资表：PAY（员工编码、岗位、绩效工资、奖金）

3. 第三范式（3NF）

第三范式（3NF）是在第二范式（2NF）的基础上建立起来的，即满足第三范式（3NF）必先满足第二范式（2NF）。第三范式要求关系表不存在非关键字列对任意候选关键字列的传递函数依赖，也就是说，第三范式要求一个关系表中不包含在其他表中已包含的非主关键字信息。

所谓传递函数依赖，就是指如果存在关键字段 A 决定非关键字段 B，而非关键字段 B 决定非关键字段 C，则称非关键字段 C 传递函数依赖于关键字段 A。

例如，这里以员工信息表（EMPLOYEE）为例，该表中包含员工编号、员工姓名、年龄、部门编码、部门经理等信息，该关系表的关键字为“员工编号”，因此存在如下决定关系：

（员工编码）→（决定）（员工姓名、年龄、部门编码、部门经理）

上面的这个关系表是符合第二范式的，但它不符合第三范式，因为该关系表内部隐含着如下决定关系：

（员工编码）→（决定）（部门编码）→（决定）（部门经理）

上面的关系表存在非关键字段“部门经理”对关键字段“员工编码”的传递函数依赖。对于上面的这种关系，可以把这个关系表（EMPLOYEE）更改为如下两个关系表。

☑ 员工信息表：EMPLOYEE（员工编码，员工姓名、年龄、部门编码）

☑ 部门信息表：DEPARTMENT（部门编码，部门经理）

对于关系型数据库的设计，理想的设计目标是按照“规范化”原则存储数据，因为这样做能够消除数据冗余、更新异常、插入异常和删除异常。

1.2.6 关系数据库的设计原则

数据库设计是指对于一个给定的应用环境，根据用户的需求，利用数据模型和应用程序模拟现实世界中该应用环境的数据结构和处理活动的过程。

数据库设计原则如下。

（1）数据库内数据文件的数据组织应获得最大限度的共享、最小的冗余度，消除数据及数据依赖关系中的冗余部分，使依赖于同一个数据模型的数据达到有效的分离。

（2）保证输入、修改数据时数据的一致性与正确性。

（3）保证数据与使用数据的应用程序之间的高度独立性。

1.3 其他常见关系数据库

关系数据库是建立在关系数据库模型基础上的数据库，通过集合代数等概念和方法来处理数据库中的数据。在这里主要介绍 Access、SQL Server 和 Oracle 数据库。

1.3.1　Access 数据库

Microsoft Access 是当前流行的关系型数据库管理系统之一，其核心是 Microsoft Jet 数据库引擎。通常情况下，安装 Microsoft Office 时选择“默认安装”，Access 数据库即被安装到计算机上。

Microsoft Access 是一个非常容易掌握的数据库管理系统。利用它可以创建、修改和维护数据库及数据库中的数据，并且可以利用向导来完成对数据库的一系列操作。Access 能够满足小型企业客户/服务器解决方案的要求，是一种功能较完备的系统，它几乎包含了数据库领域的所有技术和内容，对于初学者学习数据库知识非常有帮助。

1.3.2　SQL Server 数据库

SQL Server 数据库系列历经多次升级已经发展到 SQL Server 2012，SQL Server 是由微软公司开发的一个大型的关系数据库系统，它为用户提供了一个安全、可靠、易管理和高端的客户/服务器数据库平台。

SQL Server 是一种高性能的关系型数据库管理系统，以 Client/Server 为设计结构、支持多个不同的开发平台、支持企业级的应用程序、支持 XML 等，能够满足不同类型的数据库解决方案。SQL Server 数据库大大扩展了系统的性能、可靠性和易用性。

1.3.3　Oracle 数据库

Oracle 是美国 Oracle（甲骨文）公司提供的以分布式数据库为核心的一组软件产品。Oracle 是目前世界上使用最为广泛的关系型数据库。它具有完整的数据管理功能，包括数据的大量性、数据保存的持久性、数据的共享性、数据的可靠性等。

Oracle 在并行处理、实时性、数据处理速度方面都有较好的性能。一般情况下，大型企业选择 Oracle 作为后台数据库来处理海量数据。

1.4　DB2 的安装与配置

IBM 官方网站上提供了 DB2 Express-C 版本的软件免费下载，可以通过如下链接下载当前 DB2 Express-C 版本：http://www.ibm.com/developerworks/cn/downloads/im/udbexp/。

1.4.1　DB2 的安装

DB2 数据库的安装步骤如下。

（1）双击 setup.exe 文件，会弹出如图 1.5 所示的“DB2 安装启动板”窗口。

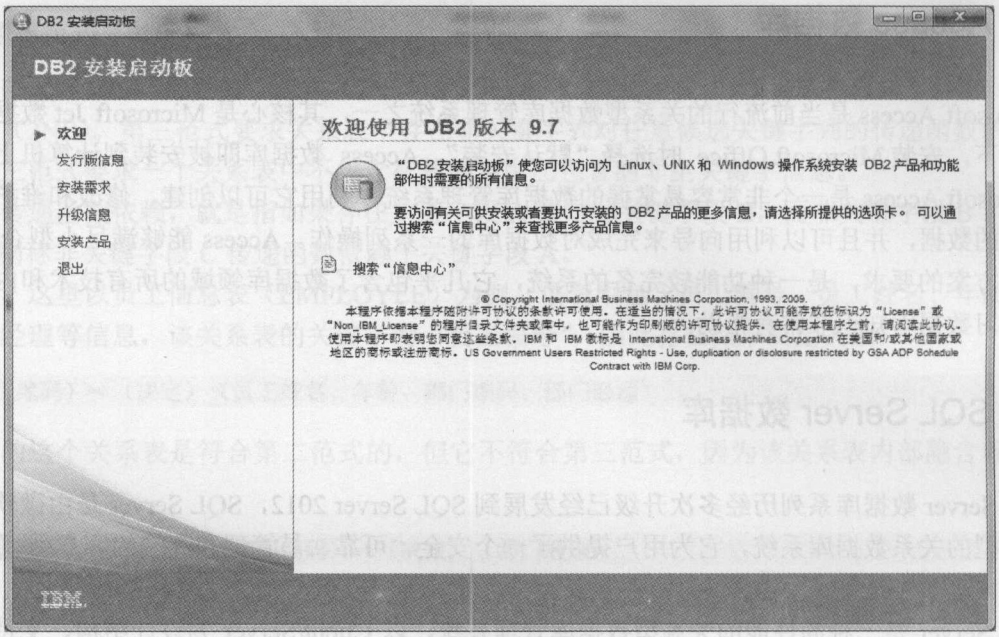


图 1.5　安装启动界面

（2）在如图 1.5 所示窗口的左侧单击“安装产品”链接，进入如图 1.6 所示的“安装产品”界面，该界面列出了所有可安装的 DB2 产品。单击“安装新产品”按钮，进入 DB2 数据库安装。

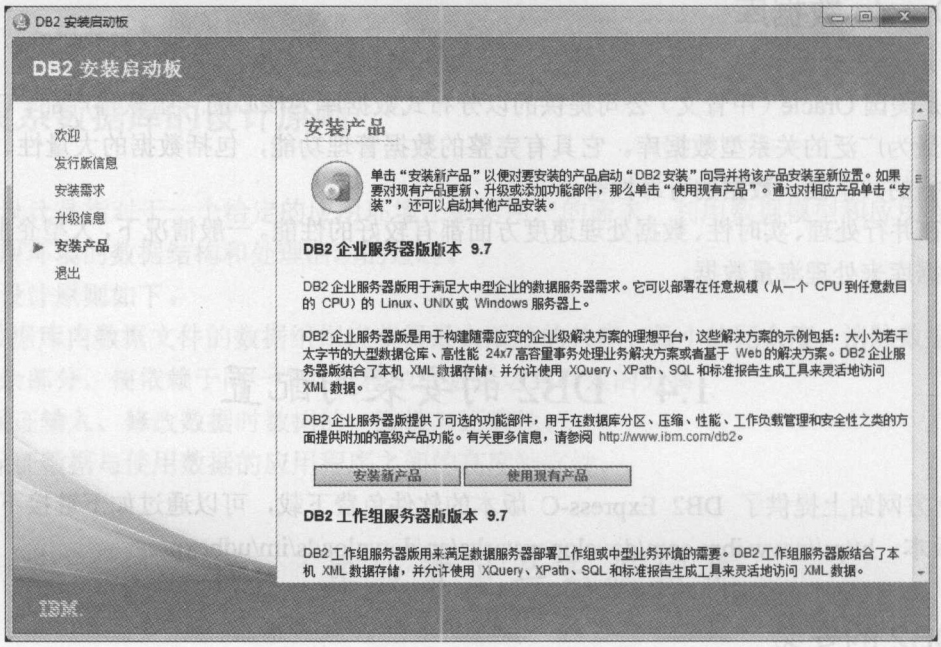


图 1.6　DB2 产品选择

（3）进入 DB2 安装后，出现如图 1.7 所示的“欢迎使用”界面，单击“下一步”按钮。

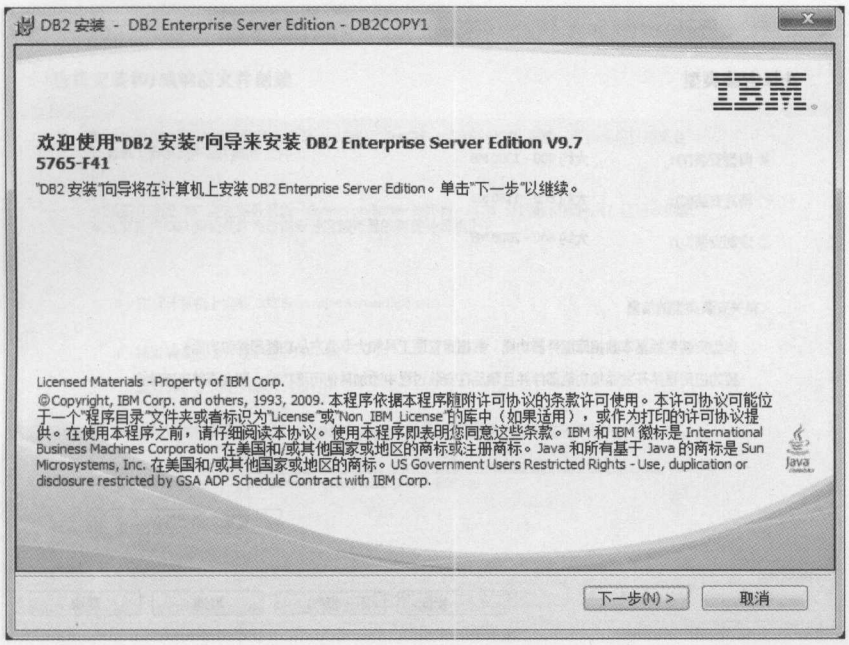


图 1.7　“欢迎使用”界面

（4）弹出“软件许可协议”界面，请认真阅读最终用户许可协议。如果接受此协议，请选中“我接受许可证协议中的全部条款”单选按钮，单击“下一步”按钮继续安装；如果不接受许可协议的条款，请单击“取消”按钮退出安装，如图 1.8 所示。

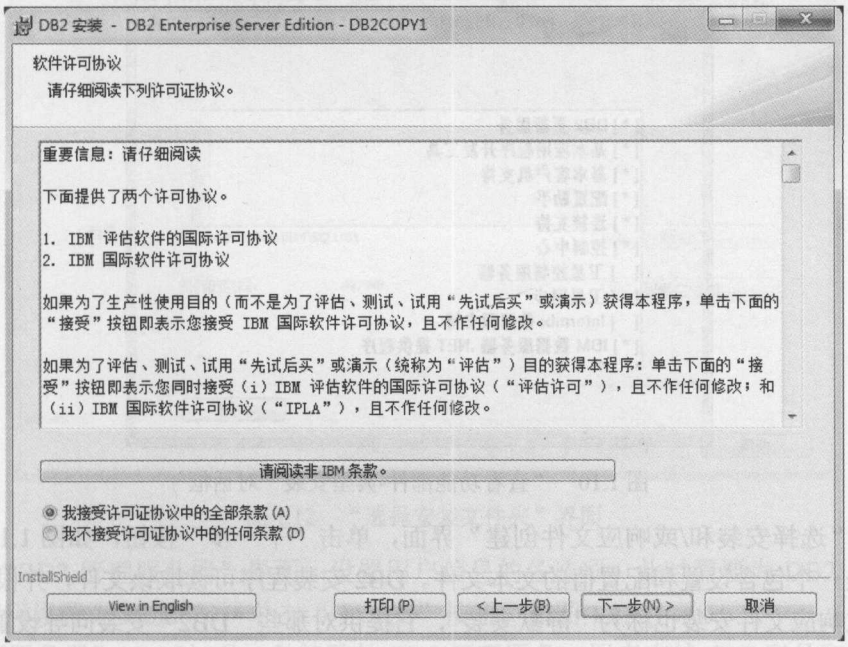


图 1.8　“软件许可协议”界面

（5）单击“下一步”按钮，弹出“选择安装类型”界面，如图 1.9 所示。

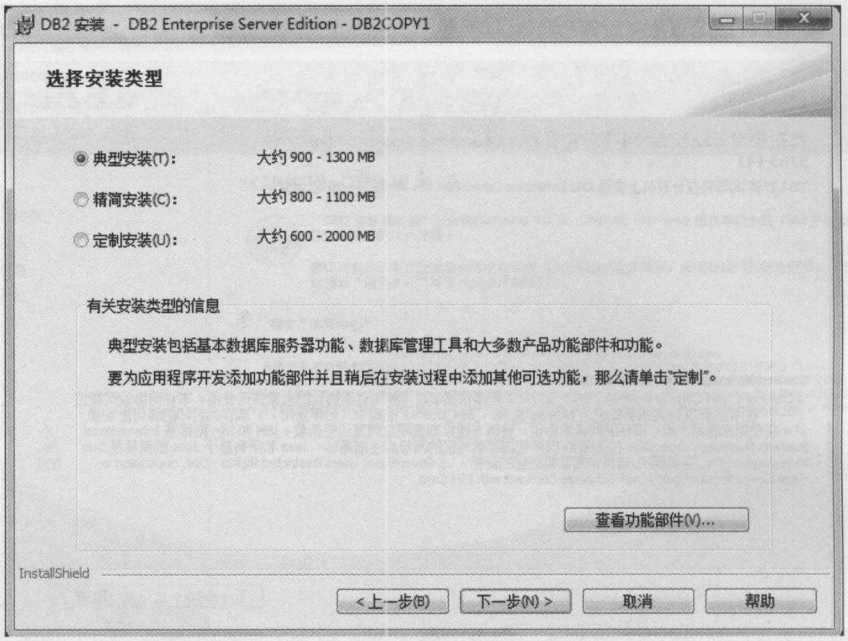


图 1.9 “选择安装类型”界面

安装类型选中“典型安装”单选按钮，通过单击该对话框的“查看功能部件”按钮，可以查看所要安装的功能部件，其中用*标记的部件为系统典型安装所配置的功能部件，如图 1.10 所示。

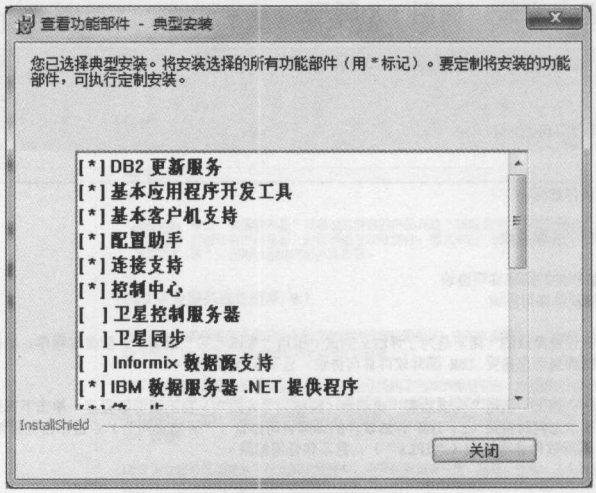


图 1.10 “查看功能部件-典型安装”对话框

（6）弹出“选择安装和/或响应文件创建”界面，单击“下一步”按钮，如图 1.11 所示。

响应文件是一个包含设置和配置值的文本文件。DB2 安装程序可读取该文件，并根据已指定的值来执行安装。通过响应文件安装也称为“静默安装”，它提供对那些“DB2”安装向导设置的参数的访问。

（7）弹出“选择安装文件夹”界面。如果按默认选项进行安装，直接单击“下一步”按钮；如果需要改变安装文件夹的路径，则单击“更改”按钮，指定安装路径，然后单击“下一步”按钮，如图 1.12 所示。

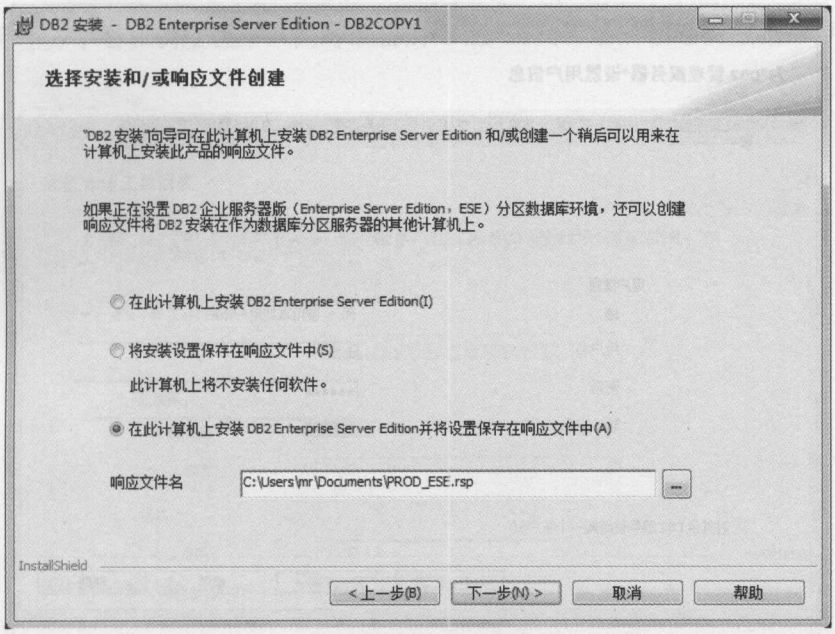


图 1.11　“选择安装和/或响应文件创建”界面

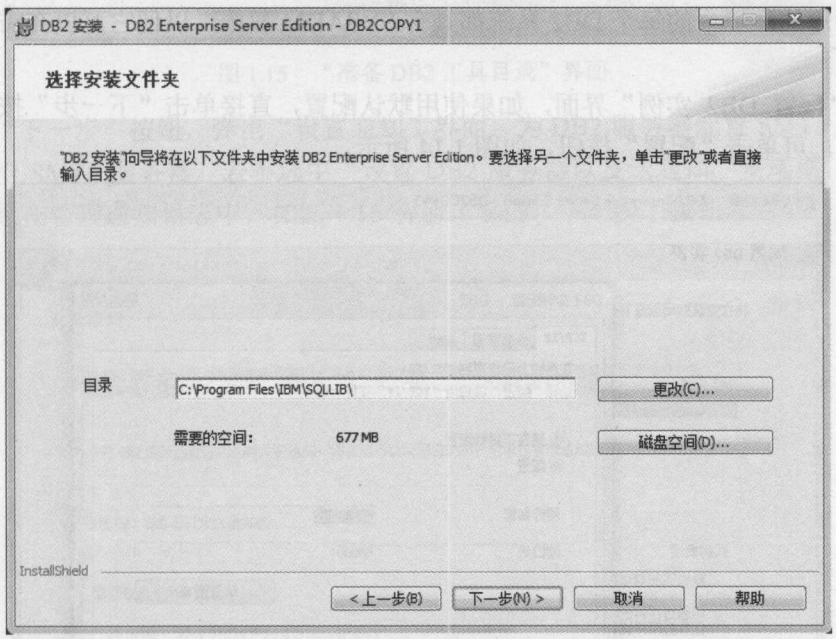


图 1.12　“选择安装文件夹”界面

（8）弹出“DB2 管理服务器”界面。设置用户信息的各选项，此时需要为 DB2 管理服务器设置密码（本书将密码设置为 ibmdb2），单击“下一步”按钮，如图 1.13 所示。

“DB2 管理服务器”（DAS）是一种特殊的 DB2 管理服务，用来支持 GUI 工具和帮助管理本地和远程 DB2 服务器上的任务。DAS 具有一个指定的用户账户，当启动 DAS 服务时，使用该用户账户来将 DAS 服务登录到计算机上。

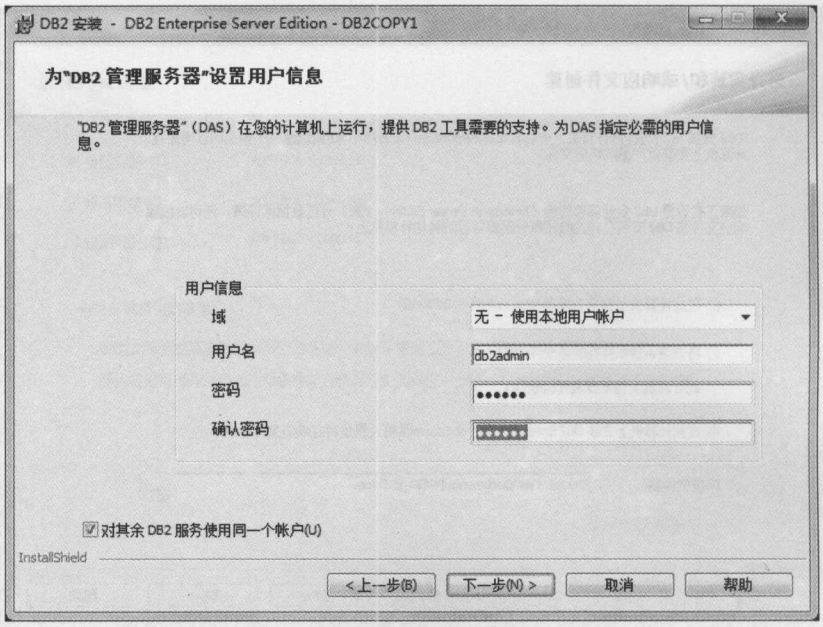


图 1.13 “DB2 管理服务器”界面

DAS 用户对您的环境中的每个 DB2 系统都具有 SYSADM 权限，以便它可以在需要时启动或停止其他实例。

（9）弹出“配置 DB2 实例”界面，如果使用默认配置，直接单击“下一步”按钮；如果要更改 DB2 实例的配置，可单击“配置”按钮，如图 1.14 所示。

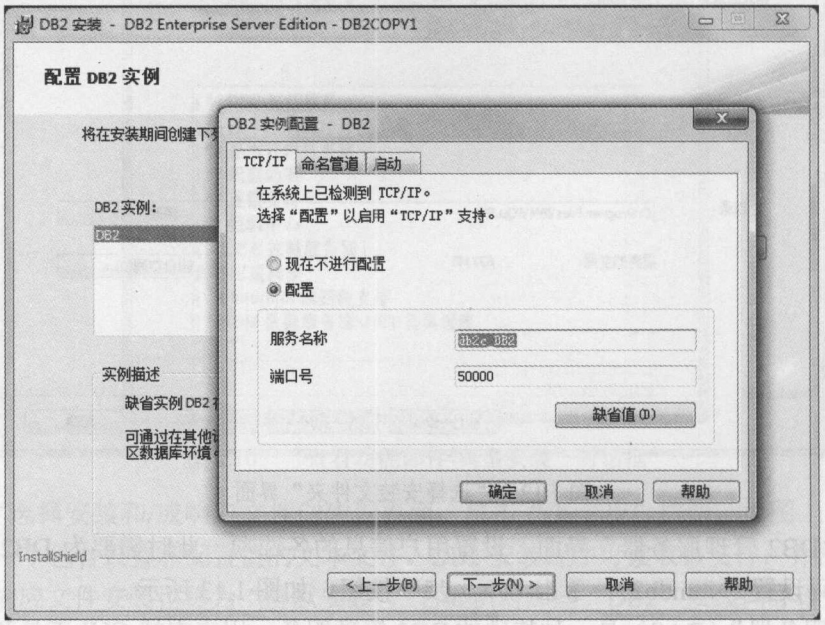


图 1.14 “配置 DB2 实例”界面

（10）弹出“准备 DB2 工具目录”界面，如图 1.15 所示。单击“下一步”按钮，只有创建 DB2

工具目录才能使用任务中心和调度程序，同时准备的 DB2 工具目录所创建的数据库存储在 DB2 数据库中。

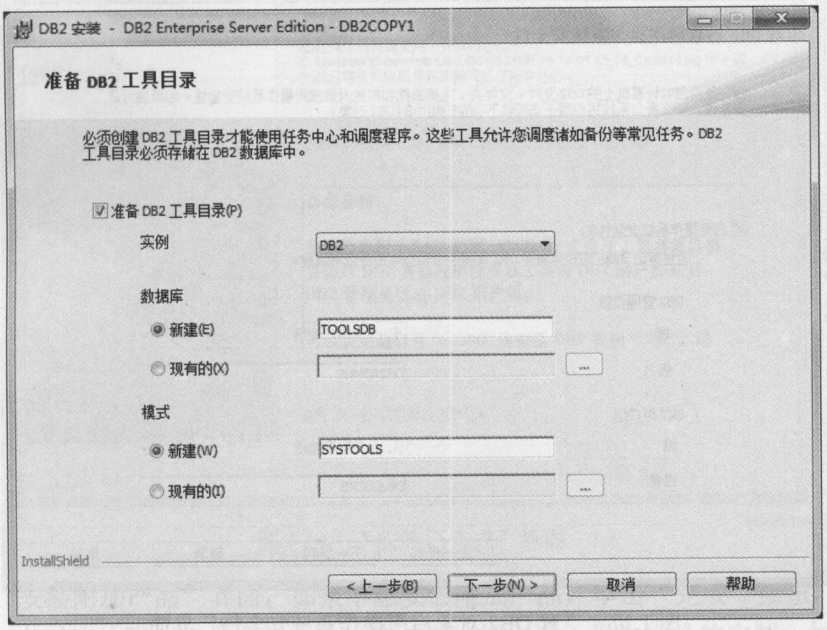


图 1.15　“准备 DB2 工具目录”界面

（11）单击“下一步”按钮，弹出“设置通知”界面，为 DB2 服务器配置 SMTP 服务器，可将运行状况警报发送给 SMTP 服务器。若不选中“设置 DB2 服务器以发送通知”复选框，则 DB2 服务器的运行状况将记录在管理通知日志中，如图 1.16 所示。单击“下一步”按钮。

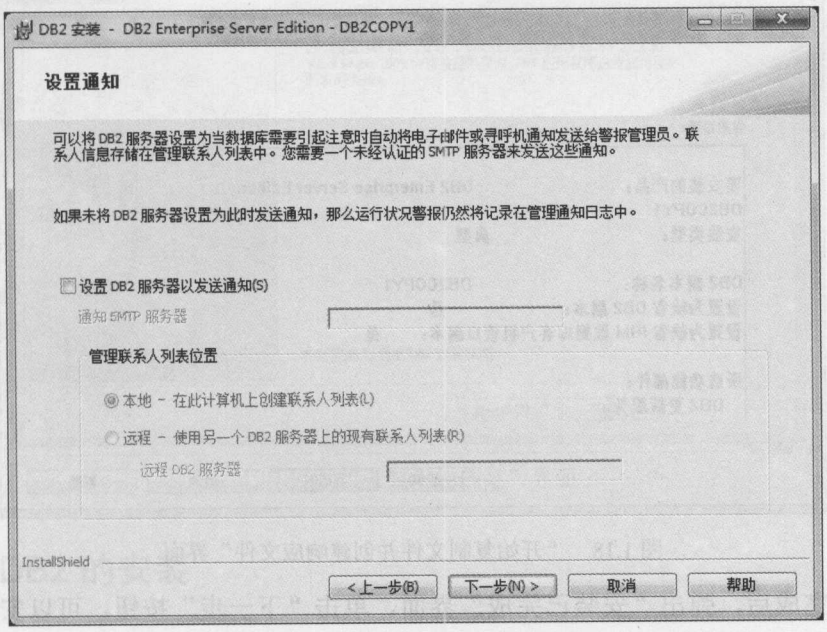


图 1.16　“设置通知”界面

（12）弹出“对 DB2 对象启用操作系统安全性”界面，如图 1.17 所示。单击“下一步”按钮。

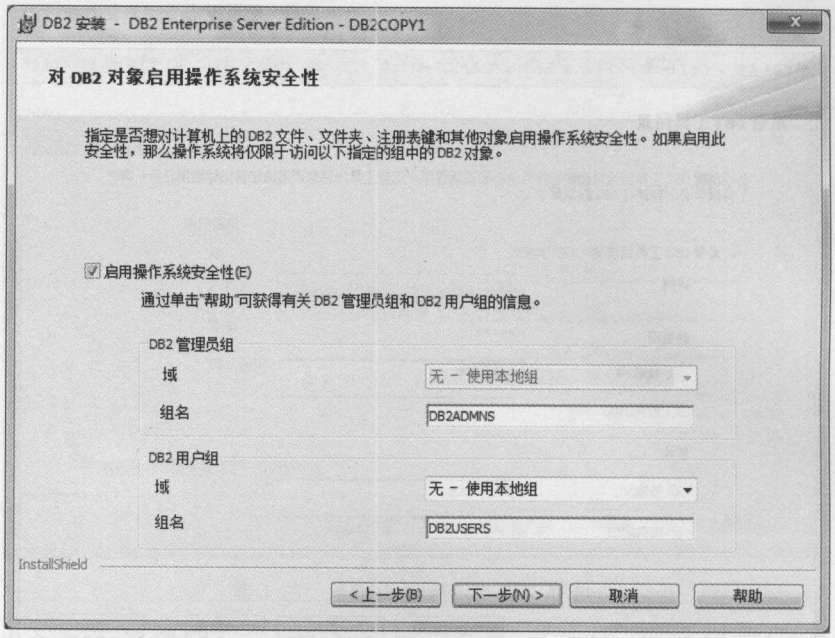


图 1.17　“对 DB2 对象启用操作系统安全性”界面

（13）弹出“开始复制文件并创建响应文件”界面，单击“完成”按钮，如图 1.18 所示。

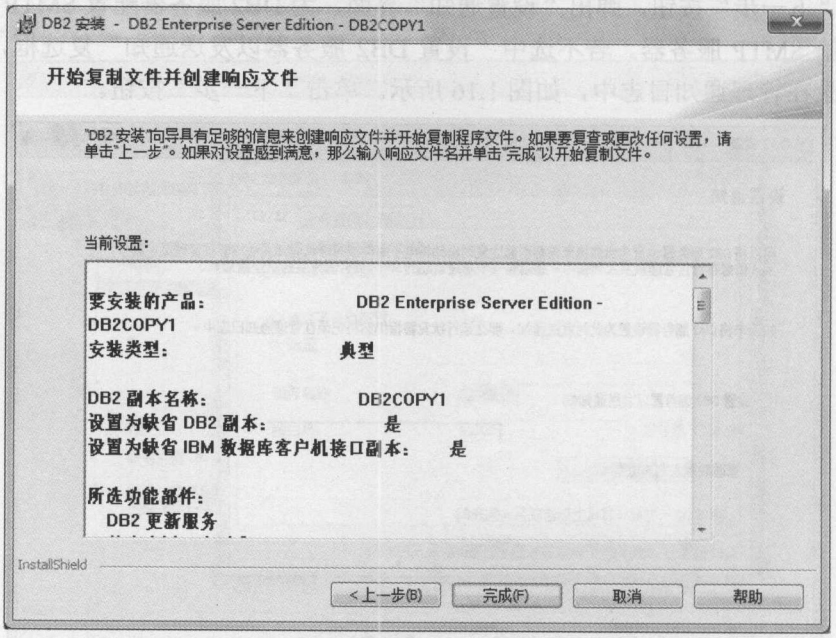


图 1.18　“开始复制文件并创建响应文件”界面

（14）安装完成后，弹出“安装已完成”界面，单击“下一步”按钮，可以安装其他产品，如图 1.19 所示。

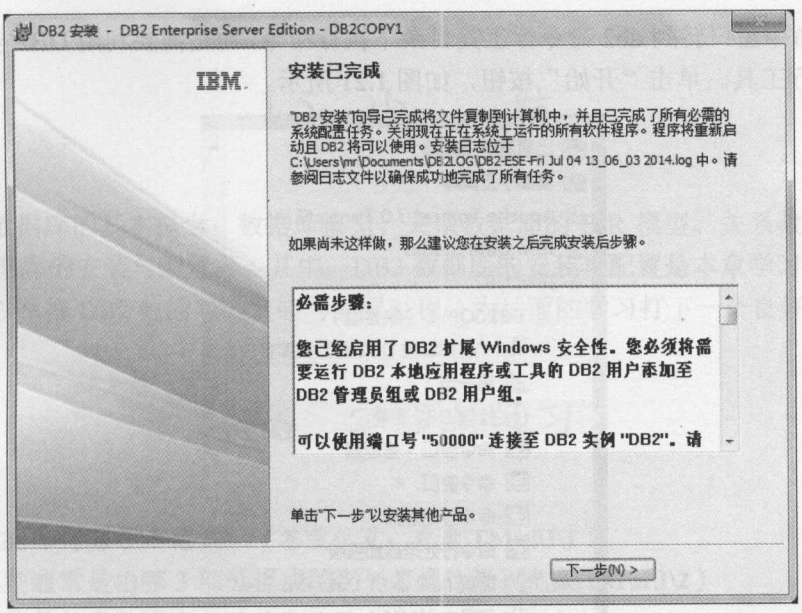


图 1.19　“安装已完成”界面

（15）弹出“安装附加产品”界面，如果不想安装附加产品，单击“完成”按钮，结束安装；如果想安装附加产品，在对应的附加产品右侧单击“安装”按钮，进入附加产品的安装，如图 1.20 所示。

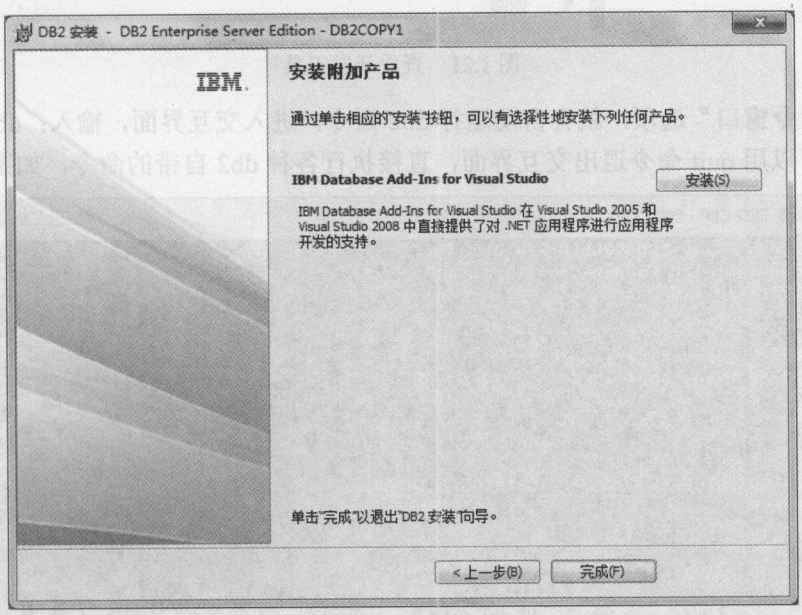


图 1.20　“安装附加产品”界面

1.4.2　验证 DB2 的安装

验证 DB2 主要是围绕命令行展开，需要注意的是，由于需要初始化一些环境变量，不能直接在

Windows 的 cmd 命令窗口转到 db2 命令行工具目录下执行命令，而是需要利用 DB2 安装的几个快捷方式执行 DB2 命令行工具。单击“开始”按钮，如图 1.21 所示。



图 1.21　查看开发工具

如果选择“命令窗口”选项，就会自动运行 db2 命令，进入交互界面，输入：db2。因为此时环境已经初始化，也可以用 quit 命令退出交互界面，直接执行各种 db2 自带的命令，如图 1.22 所示。

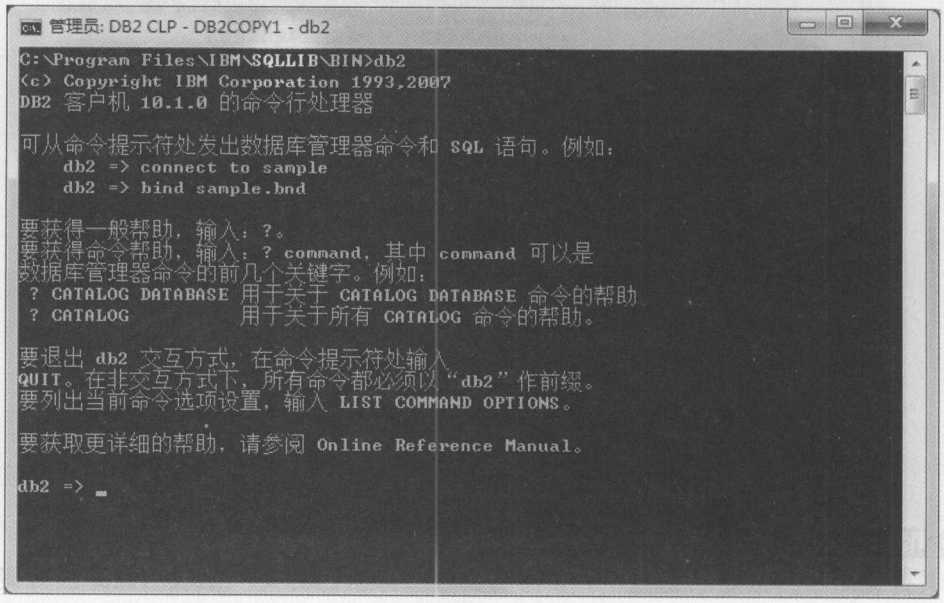


图 1.22　命令窗口

1.5　小　　结

本章介绍了数据库的基本概念：数据库模型、关系数据库的 E-R 模型、关系数据库的规范化及设计原则、DB2 数据库的安装与配置等，其中，DB2 数据库的安装与配置是本章学习的重点。通过本章的学习，读者可以掌握 DB2 数据库的安装与配置过程，为后面的学习打下一个良好的基础。

1.6　实践与练习

（1）关系数据库有哪 3 个范式？（**答案位置：光盘\TM\sl\1\1**）

（2）数据模型通常是由哪 3 部分组成的？（**答案位置：光盘\TM\sl\1\2**）

（3）下面哪些是关系数据库？（**答案位置：光盘\TM\sl\1\3**）

① Access　　② SQL Server

③ Oracle　　④ XML

第 2 章

DB2 的体系结构与 DB2 工具

（视频讲解：66 分钟）

本章主要介绍 DB2 的体系结构。与其他关系数据库产品相似，DB2 主要提供数据存储和数据访问能力。本章将通过 DB2 数据库对象层次和数据动态访问过程来讲述 DB2 的体系结构，并对 DB2 的命令工具和其他一些常用工具进行介绍。

通过阅读本章，您可以：

- 熟悉 DB2 的体系结构
- 了解 DB2 对象层次关系
- 了解数据访问过程
- 掌握 DB2 命令工具的使用
- 掌握配置助手的使用
- 熟悉控制中心
- 熟悉工具设置
- 熟悉向导

2.1　DB2 体系结构

DB2 数据库主要包括系统、实例、环境变量、配置参数、数据库和表空间等内容，下面分别进行介绍。

1．系统

DB2 体系结构中的最高一层是系统，一个系统表示 DB2 的一个安装。当前的应用系统主要分为两类：联机事务处理（OLTP）和联机分析处理（OLAP）。OLTP 主要执行日常的事务处理，例如银行存取款、商场买东西等，它的主要特点是对响应时间要求高，数据量一般较小，并发多，面向应用。OLAP 主要指数据仓库、决策分析类系统，主要特点是数据量大，对实时性要求不高，面向主题。在由很多机器组成的网络环境中，我们有时也称系统为数据库分区。一个系统可以包含多个 DB2 实例，每个实例能够管理一个或多个数据库。

2．实例

实例也称为数据库管理器（Database Management Application），是数据库管理器在内存中的映像，是管理数据的 DB2 代码。实例相当于 Informix 的 Informix Server，在一台机器上可以有多个相互独立的实例，实例之间彼此独立，同时运行，不会相互影响。每个实例可以管理若干个数据库，一个数据库只属于一个实例。

3．环境变量

环境变量是在操作系统级别上定义的变量。例如，在 Windows 平台中，可以为变量创建新的项，也可以通过“控制面板”→“系统”→“高级”→“环境变量”来编辑现有变量的值。DB2 通过使用概要文件注册变量来集中控制环境变量，不同的概要文件提供了不同级别的支持。注册变量定义了 DB2 操作环境，这些变量存储在 DB2 注册文件中，分为两种注册变量。一种是全局变量，变量的设置是系统范围的。另一种是实例变量，变量的设置用于特定的实例。

4．配置参数

配置参数是在不同的级别上定义的。每个级别上的变量都是不同的。配置参数能够影响数据库或数据库管理员的操作特性，它们存储在配置文件中。许多参数都会被数据库设置为默认值，管理员可以根据自己的需要修改它们。数据库的配置文件是在数据库创建的时候建立的，位于数据库所在的目录。数据库的配置参数定义了分配给数据库的各种资源的数量。参数值可以被修改，以满足数据库运行的各种需求。

5．数据库

关系数据库是使用一组表来管理数据的，表是由在行和列中以逻辑关系排列的数据组成，每个表的数据在逻辑上相关，在表之间能够定义关系。每个数据库中都包含一组系统编目表、配置文件和恢复日志。系统编目表用于描述数据的逻辑和物理结构，配置文件包含所有为数据库分配的配置参数值，

恢复日志记录正在进行的事务处理和可存档的事务处理。

6. 表空间

表空间是数据库中表数据与数据库之间的逻辑中间层，数据库中的物理空间组织为表空间的集合，而表空间是表的逻辑集合。表空间包含容器集合，容器是用来描述物理空间分配的一般术语。表存储在一个或几个表空间中，可以将表中不同类型的数据分别存储在不同的表空间中，以便提高性能或方便表空间的备份。表空间最终会映射到物理存储介质上，对物理存储的合理使用可以让管理员有效地控制数据库的性能。

使用表空间的一个明显好处是能够把数据合理地分布存储在不同的磁盘上或者存储在磁盘的不同位置上，有助于提高数据存取的效率。

2.2 对象层次关系

数据库对象逻辑层中最高层的对象是系统。一个系统可以创建一个或多个实例，每个实例可以管理一个或多个数据库。关于实例和数据库的关系很多初学者很容易混淆。此处可以通过一个比喻来理解。数据库好比 ATM 机，而实例就是银行卡，存取钱必须要有银行卡，即使银行卡坏了，ATM 机仍然在，但却无法使用 ATM 机。对应到 DB2 的术语，只有实例在运行的时候，才可以访问数据库。如果实例挂起或停止了，虽然数据仍在，但却无法访问。

在 DB2 中，一个实例可以包含多个数据库，删除实例并不会删除数据库，但一个数据库只能属于一个实例。

表是数据库最重要的对象，表是二维结构，由行和列构成。对数据库的访问和操作，是通过表来实现的。表空间的作用是用来管理表数据。

可以把整个数据库想象成一个 N 层的大楼，每一层就相当于一个表空间。当用户创建一个表的时候，都需要指定创建在哪个表空间中，也就是相当于去大楼的哪一层。

每一个表空间可以容纳一个或多个表与索引等对象。表空间实际上是逻辑上的概念，物理上对应着一个或多个容器。容器才是真正用来存放数据的地方，可以是文件，也可以是没有格式化过的磁盘，还可以是目录。一个表空间可以跨越多个容器，但每个容器只能归属于一个表空间。举个例子，容器可以看成是一幢大楼里每一层中的各个单元。

DB2 的对象层次关系相信大家已经有了一个初步认识。下面总结一下，就是一个系统可以创建一个或多个实例，每一个实例中包含一个或多个数据库，每个数据库中包含一个或多个表空间，每个表空间包含一个或多个容器，而每个表和索引则可以存在于一个或多个表空间中。

2.3 数据访问过程

一个客户端在访问数据前，首先要知道访问的是哪台机器上的哪个数据库，这个操作叫作编目。

编目就是在客户端做个注册，把注册机器的动作叫作节点编目，把注册数据库的动作叫作数据库编目。

数据库编目成功，客户端就可以通过用户名、密码连接数据库了，当编目信息与认证信息都正确时，数据库连接就会被建立。从数据库的角度看，每一个数据连接就是一个应用程序。如果客户端使用同一个可执行文件同时执行 10 次，尽管每次执行的文件都相同，但是从数据库的角度看，依然是接收 10 个连接请求，也就是 10 个应用程序，而不是一个。

每一个应用程序在同一时刻只能运行一个事务。事务是一组要么全成功要么全失败的操作。例如从银行取钱，第一步我们的银行账户的金额会减少，第二步我们的现金会增多，如果第二步失败了（例如银行账户中的钱小于取款金额），那么我们要撤销第一步操作中银行账户金额数量的更改，也就是让两步操作同时失败。同理，如果第二步操作成功了，那么我们提交这个事务，让数据修改生效。

在每一个事务中，可以包含一个或多个语句。其中有的语句可以用来插入数据，有的语句可以用来更新或者查询数据。当该事务中所有的操作都执行完毕之后，用户可以提交（Commit）或者回滚（Rollback）事务，来确保数据的完整性。

当客户端发起一个语句，应用程序会使用某种驱动（CLI、JDBC 或者 ODBC）与 DB2 服务器交流，当一个语句被驱动拿到后，就会被打成一个网络包发送给数据库的接收端。当服务器端接收到这个包时，DB2 会根据这个包的数据结构解析出 SQL 语句。然后 DB2 编译器对这个语句进行解析，并通过优化器得到执行计划。所谓的执行计划就是一组告诉 DB2 如何运行一个查询的数据结构。然后数据库运行时会根据执行计划生成一个包。当执行这个包时，DB2 引擎就可以从磁盘上相应位置获取数据，或者与其他的表进行关联。当获得结果后，通过先前建立的 TCP/IP 连接把结果发送到客户端。

总的来说，数据访问流程就是客户端首先连接数据库，每个连接相当于一个应用程序，这个应用程序由一个或者几个代理负责完成。每个应用程序顺序执行一些事务处理，每个事务由一条或多条 SQL 语句组成。

2.4　DB2 命令工具的使用

DB2 数据库的命令工具主要有命令编辑器、命令窗口和命令行处理器等 3 种，本节将对它们的使用及区别进行详细讲解。

2.4.1　命令编辑器

通过 4 种方式可以打开“命令编辑器”对话框，如图 2.1 所示。

- ☑ 选择“开始”→“所有程序”→IBM DB2→“DB2COPY1（默认值）”→“命令行工具”→“命令编辑器”命令。
- ☑ 按下 Windows+R 快捷键，在弹出的“运行”对话框中输入 db2ce，然后单击“确定”按钮。
- ☑ 在“控制中心”、“配置助手”、“运行状况中心”、“复制中心”、“任务中心”或“日志”窗口的主菜单中选择“工具”菜单项，在列出的子菜单中选择“命令编辑器”菜单项。
- ☑ 在“控制中心”、“配置助手”、“运行状况中心”、“复制中心”、“任务中心”或“日志”窗口的

工具栏单击按钮。

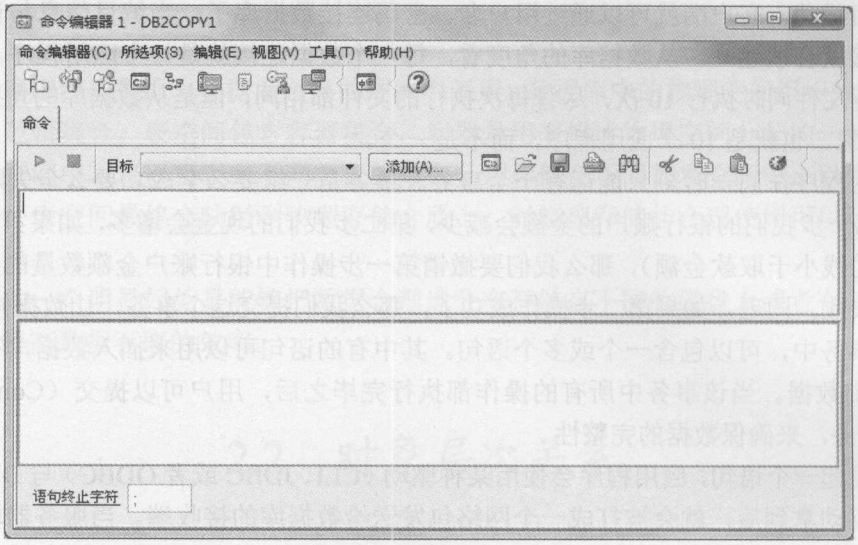


图 2.1 “命令编辑器”对话框

命令编辑器可以执行 SQL 语句，查看查询结果，允许用户查看 SQL 语句的存取计划。用户可以通过“命令编辑器”为 SQL 语句生成存取计划，并以可视化的形式表现出来。

2.4.2 命令窗口

用下面任意一种方法可以打开“命令窗口”，“命令窗口”是 DB2 非交互方式下的命令窗口，如图 2.2 所示。

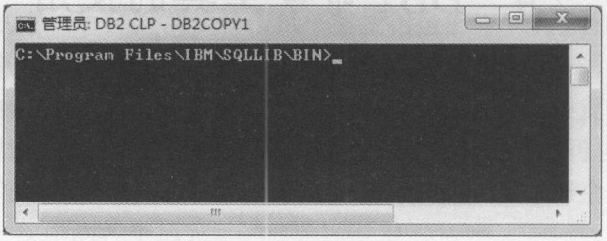


图 2.2 命令窗口

- ☑ 选择“开始”→“所有程序”→IBM DB2→“DB2COPY1（默认值）”→“命令行工具”→“命令窗口”命令。
- ☑ 按下 Windows+R 快捷键，在弹出的“运行”对话框中输入 db2cw，然后单击“确定”按钮。

2.4.3 命令行处理器

“命令行处理器”是 DB2 交互方式下的窗口。使用下面的方法能够启动“命令行处理器”窗口。

选择“开始”→“所有程序”→IBM DB2→“DB2COPY1（默认值）”→“命令行工具”→“命令行处理器”命令，窗口如图 2.3 所示。

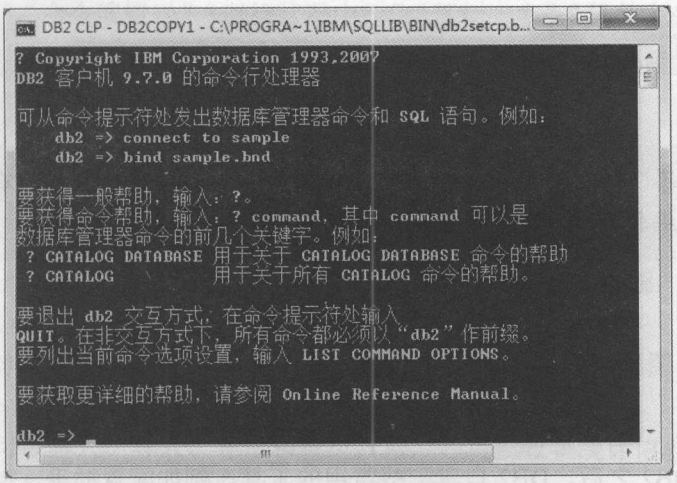


图 2.3　命令行处理器

2.4.4　“命令编辑器”“命令窗口”“命令行处理器”用法的区别

虽然“命令编辑器”、“命令窗口”和“命令行处理器”都可以执行 SQL 语句，但是它们在书写 SQL 语句的格式上还是有一定区别的。

（1）在“命令行处理器”中执行 SQL 命令时，每个 SQL 语句末尾不可以加分号（;），否则会报错，系统认为分号是异常标记，且一次可执行多条 SQL 语句组成的命令（系统认为回车符或换行符是语句结束符，分号不能作为语句结束符）。

例如：

```
CONNECT TO MRKJ
SELECT * FROM mr.商品详细信息表 WHERE 商品名字 = '手机'
```

（2）在“命令窗口”中执行 SQL 命令时，每个 SQL 语句开始位置必须加语句开始标记“db2”或“DB2”（注意：在“db2”或“DB2”后面都要有空格），且一次可执行多条 SQL 语句组成的命令（系统认为回车符或换行符是语句结束符，分号不能作为语句结束符）。

例如：

```
DB2 CONNECT TO MRKJ
DB2 SELECT * FROM mr.商品详细信息表 WHERE 商品名字 = '手机'
```

或

```
db2 CONNECT TO MRKJ
db2 SELECT * FROM mr.商品详细信息表 WHERE 商品名字 = '电视'
```

（3）在“命令编辑器”中执行 SQL 命令时，只有分号（;）会被系统认为是语句的结束符，回车

符或换行符不是语句结束符。如果命令中只包含一条 SQL 语句，则语句结束符可以省略。

例如：

```
CONNECT TO MRKJ;
SELECT * FROM mr.商品详细信息表 WHERE 商品名字 = '手机'
```

注意

由于“命令编辑器”图形界面能够友好显示而且含有较多控件，如果没有特殊说明，本书中所有的命令都在“命令编辑器”中执行。

2.5 DB2 常用工具

在计算机上安装 DB2 之后，DB2 自带一些常用的工具，例如配置助手、控制中心、向导工具等，本节就对 DB2 的一些常用工具进行讲解。

2.5.1 配置助手

“配置助手”窗口如图 2.4 所示，启动配置助手的几种方式如下：

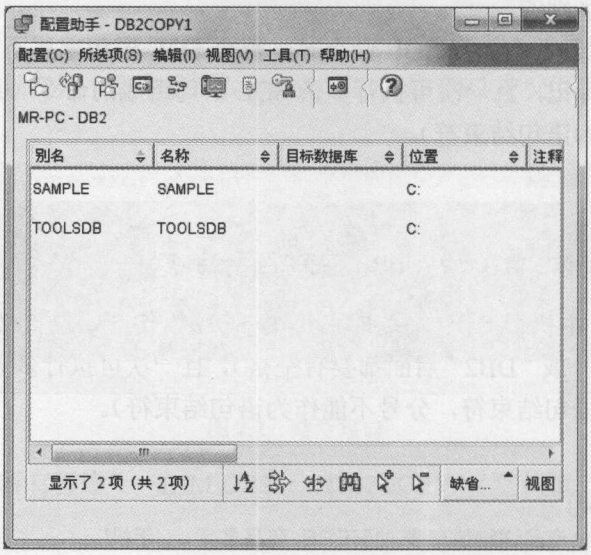


图 2.4 “配置助手”窗口

- ☑ 选择“开始”→“所有程序”→IBM DB2→“DB2COPY1（默认值）”→“设置工具”→“配置助手”命令。
- ☑ 按下 Windows+R 快捷键，会弹出“运行”对话框。在“运行”对话框中输入 db2ca，然后单击“确定”按钮。

☑ 在“控制中心”“命令编辑器”“运行状况中心”“复制中心”“任务中心”“日志”窗口的主菜单中选择“工具”菜单项，在列出的子菜单中选择“配置助手”菜单项。

☑ 在“控制中心”“命令编辑器”“运行状况中心”“复制中心”“任务中心”“日志”窗口的工具栏中单击按钮。

2.5.2　控制中心

可以使用“控制中心”来执行以下一些重要任务：

（1）将 DB2 数据库系统、联合系统、DB2 UDB z/OS 和 OS/390 版系统、IMSysplexes、实例、数据库和数据库对象添加至对象视图。

（2）管理数据库对象。可以创建、改变和删除数据库、表空间、表、视图、索引、触发器和模式，还可以管理用户。

（3）管理数据。可以装入、导入、导出和重组数据，还可以收集统计信息。

（4）通过备份和恢复数据库或表空间来执行预防性维护。

（5）配置和调整实例和数据库。

（6）管理数据库连接，例如，DB2 Connect 服务器和子系统。

（7）管理 IMS 系统。

（8）管理 DB2 UDB z/OS 和 OS/390 版子系统。

（9）管理应用程序。

（10）使用 Visual Explain 分析访问方案。

1. 控制中心的 4 种启动方式

☑ 选择“开始”→“所有程序”→IBM DB2→“DB2COPY1（默认值）”→“一般工具”→“控制中心”命令。

☑ 按下 Windows+R 快捷键，在弹出的“运行”对话框中输入 db2cc，然后单击“确定”按钮。

☑ 在“配置助手”、“命令编辑器”、“运行状况中心”、“复制中心”、“任务中心”或“日志”窗口的主菜单中选择“工具”菜单项，在列出的子菜单中选择“控制中心”菜单项。

☑ 在“配置助手”、“命令编辑器”、“运行状况中心”、“复制中心”、“任务中心”或“日志”窗口的工具栏中单击按钮。

2. “控制中心视图”的 3 个选项

在“控制中心视图”对话框中，有“基本”、“高级”和“定制”3 个选项，如图 2.5 所示。

它们的功能如下所述。

☑ 基本：此选项提供核心 DB2 数据库功能，它包括一些基本对象，例如，数据库、表和存储过程等。

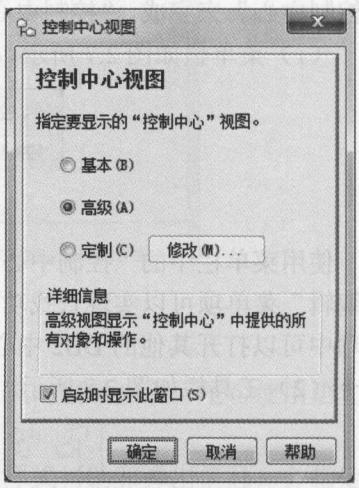


图 2.5　“控制中心视图”对话框

☑ 高级：此选项显示“控制中心”中提供的所有对象和操作。

☑ 定制：此选项使用户能够定制对象树和对象操作以满足特定需要。

本书选择的是“高级”选项，选择好控制中心视图种类后，单击“确定”按钮，即可打开“控制中心”窗口，如图 2.6 所示。

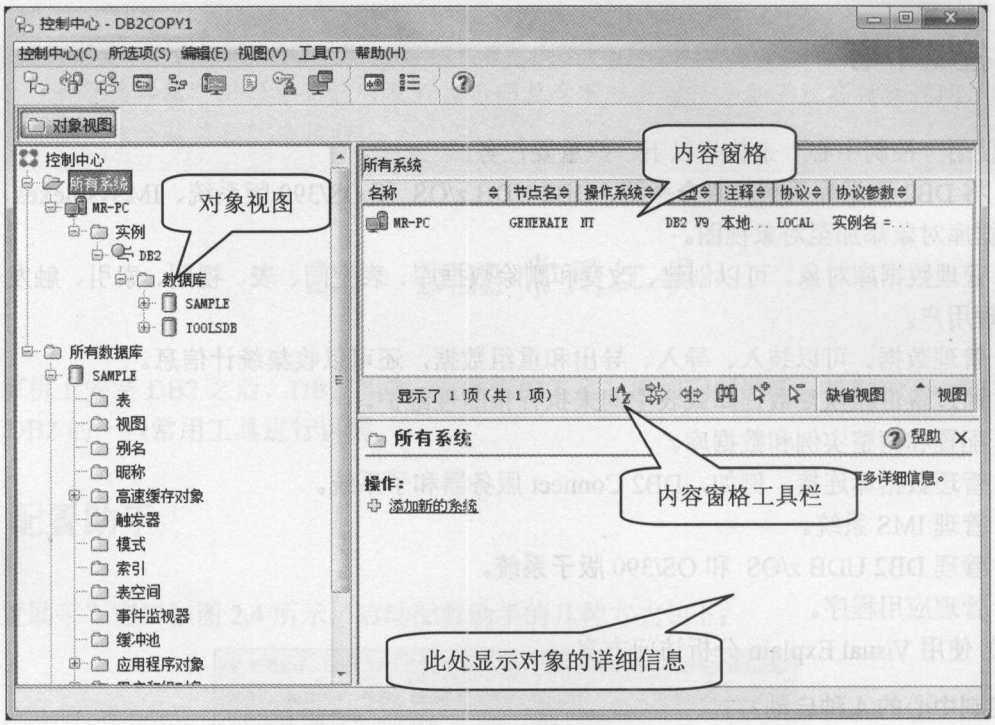


图 2.6 “控制中心”窗口

3. “控制中心”窗口的组成

“控制中心”是 DB2 管理工具的核心，绝大多数的管理任务和对其他管理工具的存取都可以通过“控制中心”来完成。“控制中心”窗口包含以下几个部分。

（1）菜单栏如图 2.7 所示。

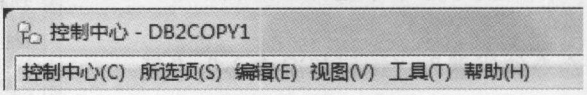


图 2.7 “控制中心”菜单栏

使用菜单栏中的“控制中心”菜单项可以关闭 DB2 工具，“所选项”菜单项可以创建定制文件夹，“编辑”菜单项可以实现查找功能，“视图”菜单项可以实现新建、过滤、排序等功能，在“工具”菜单项中可以打开其他的 DB2 中心和工具设置，“帮助”菜单项中包含一些联机帮助。

（2）工具栏如图 2.8 所示。



图 2.8 “控制中心”工具栏

（3）对象视图：使用对象视图来显示和处理文件夹和文件夹对象。任意选择一项将在内容窗格和对象详细信息窗格中显示相关对象、操作和信息。在其中的一项上单击鼠标右键，将弹出快捷菜单，该菜单列出了可对该项执行的所有操作。想要展开或折叠对象视图，可以单击视图中对象旁的加号（+）和减号（-）。

（4）内容窗格：用来显示在对象视图中选中的文件夹包含的所有对象。选择某对象，将在“对象详细信息”窗格中显示与该对象相关联的操作和信息。

（5）“对象详细信息”窗格：用来显示在内容窗格中选中的对象信息，如果未显示该窗格，则从“控制中心”主菜单中选择“视图”菜单项，在列出的子菜单中选择“显示对象详细信息窗格”菜单项。

（6）内容窗格工具栏：使用内容窗格工具栏来定制内容中的对象和信息视图，可以适应用户的需要。

2.5.3　工具设置

“工具设置”用来为 DB2 管理工具定制设置和设置属性，包括文档设置、字体和颜色设置等。通过以下两种方法可以打开“工具设置”窗口：

☑ 通过单击“控制中心”工具栏上的图形按钮来启动。

☑ 在“控制中心”主菜单中选择“工具”菜单项，然后选择“工具设置”命令。

“工具设置”窗口如图 2.9 所示。

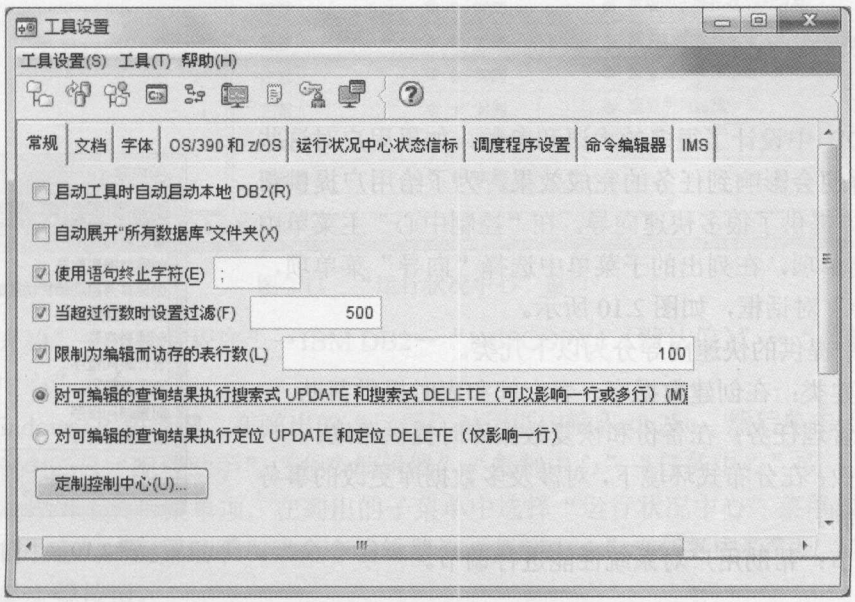


图 2.9　“工具设置”窗口

工具设置窗口中有以下几个选项卡。

☑ 常规：此选项卡用来指定本地 DB2 实例是否应在启动 DB2 工具时自动启动，是否使用语句终止字符以及显示样本内容请求超过最大行数时是否设置过滤。

☑ 文档：此选项卡用来指定是否在 DB2 管理工具中自动显示悬浮式帮助和弹出信息，并指定在

实例级别访问上下文帮助的位置。

- ☑ 字体：此选项卡用来更改在 DB2 管理工具中显示的文本和菜单的字体。从可用的字段中选择想要在 DB2 管理工具中显示的菜单的字体大小和颜色。
- ☑ OS/390 和 z/OS：此选项卡用来设置列标题，并定义 OS/390 和 z/OS 对象的联机和批处理实用程序执行选项。某些选项提供了默认值。
- ☑ 运行状况中心状态信标：此选项卡用来指定在“运行状况监视器”中生成警报时将接收到的通知的类型。可以通过弹出消息或使用图形信标来接受通知，或是同时使用上述两种方法接收通知。
- ☑ 调度程序设置：此选项卡用来设置默认调度方案。如果希望任务调度要由数据库服务器本地的调度程序来处理，则在“默认调度方案”栏中选择“服务器调度”选项。如果希望任务的存储和调度由集中系统来处理，则在“默认调度方案”中选择“集中调度”选项，在这种情况下需要从“集中调度程序”的列表框中选择某个集中系统。
- ☑ 命令编辑器：此选项卡用来指定如何生成、编辑、执行、处理 SQL 和 XQuery 语句、IMS 命令和 DB2 命令以及处理产生的输出。这些设置影响 DB2 数据库、在 z/OS 和 OS/390 系统与子系统以及 IMSysplexes 上的命令、SQL 语句和 XQuery 语句。
- ☑ IMS：此选项卡用来设置使用 IMS 时的首选项。可以设置语法支持、启动可用的向导、显示 IMS 结果和 IMS 结果历史的大小。

2.5.4 向导

在 DB2 UDB 中设计了很多的术语和参数，如果用户对这些信息不是很了解便会影响到任务的完成效果。为了给用户提供帮助，DB2 UDB 中提供了很多快速向导。在“控制中心”主菜单中选择“工具”菜单项，在列出的子菜单中选择“向导”菜单项，便可打开“向导”对话框，如图 2.10 所示。

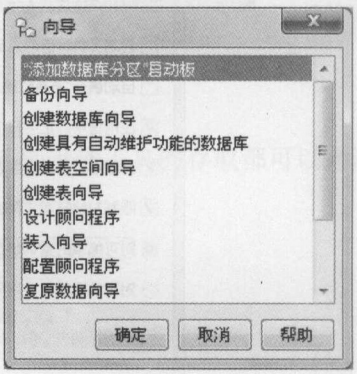


图 2.10 “向导”对话框

DB2 UDB 中提供的快速向导分为以下几类。

- ☑ 对象创建类：在创建数据库、表空间和表时提供帮助。
- ☑ 数据库管理任务：在备份和恢复数据库时提供帮助。
- ☑ 多点更改：在分布式环境下，对涉及多数据库更改的事务进行控制。
- ☑ 性能调节：帮助用户对系统性能进行调节。

2.5.5 信息中心

通过以下两种方式可以打开“信息中心”窗口：

- ☑ 选择“开始”→“所有程序”→IBM DB2→“DB2COPY1（默认值）”→“信息”→“信息中心 V9.7”命令。

☑ 按下 Windows+R 快捷键，在弹出的“运行”对话框中输入 db2ic，然后单击“确定”按钮。

从“信息中心”可以获取几乎所有信息，包括“DB2 解决方案信息中心主页”、“产品概述”、“数据库基本内容”、“数据库管理”、“数据库应用程序开发”、“数据仓储和分析”、Federation、Replication and event publishing、“Spatial 和地理数据”以及“词汇表”等极其丰富的内容。

2.5.6　运行状况中心

可以用下面任何一种方式启动“运行状况中心”窗口，窗口如图 2.11 所示。

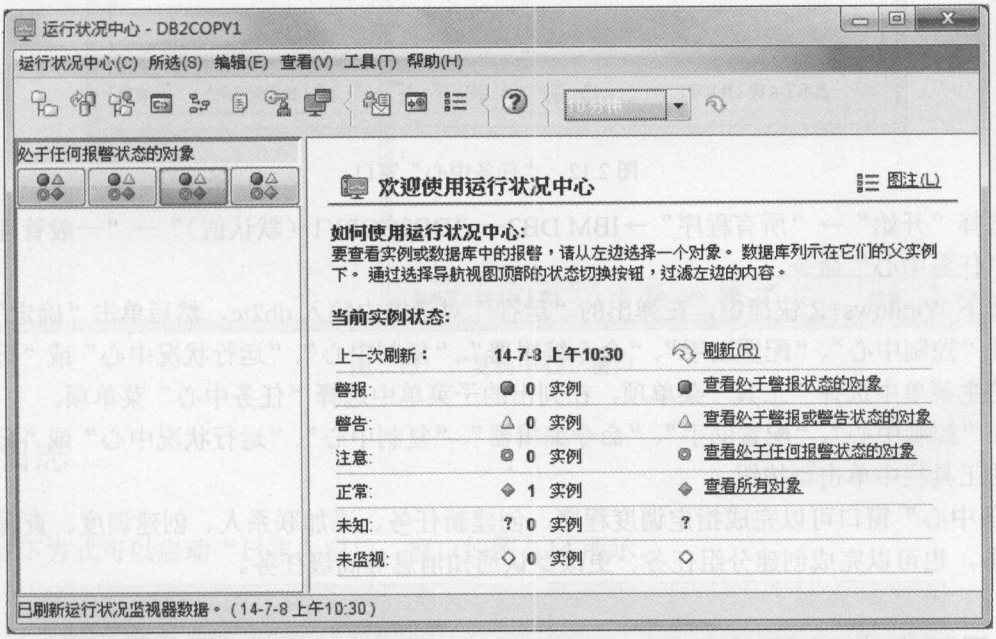


图 2.11　“运行状况中心”窗口

☑ 选择“开始”→“所有程序”→IBM DB2→“DB2COPY1（默认值）”→“监视工具”→“运行状况中心”命令。

☑ 按下 Windows+R 快捷键，在弹出的“运行”对话框中输入 db2hc，然后单击“确定”按钮。

☑ 在“控制中心”、“配置助手”、“命令编辑器”、“复制中心”、“任务中心”或“日志”窗口的主菜单中选择“工具”菜单项，在列出的子菜单中选择“运行状况中心”菜单项。

☑ 在“控制中心”、“配置助手”、“命令编辑器”、“复制中心”、“任务中心”或“日志”窗口的工具栏中单击按钮。

2.5.7　任务中心

通过以下方式可以启动“任务中心”，窗口如图 2.12 所示。

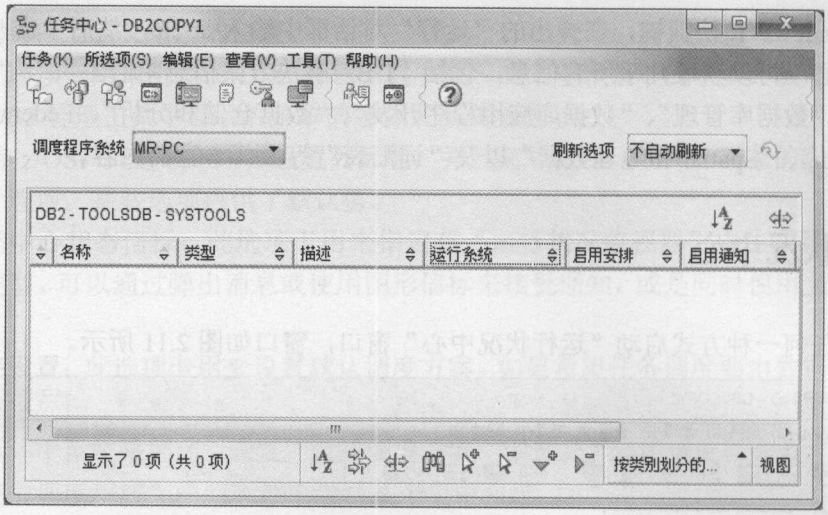


图 2.12 “任务中心”窗口

- ☑ 选择“开始”→“所有程序”→IBM DB2→“DB2COPY1（默认值）”→“一般管理工具”→“任务中心”命令。
- ☑ 按下 Windows+R 快捷键，在弹出的“运行”对话框中输入 db2tc，然后单击“确定”按钮。
- ☑ 在“控制中心”、“配置助手”、“命令编辑器”、“复制中心”、“运行状况中心”或“日志”窗口的主菜单中选择“工具”菜单项，在列出的子菜单中选择“任务中心”菜单项。
- ☑ 在“控制中心”、“配置助手”、“命令编辑器”、“复制中心”、“运行状况中心”或“日志”窗口的工具栏中单击按钮。

“任务中心”窗口可以完成指定调度程序、创建新任务、添加联系人、创建调度、查看任务进度等基本任务，也可以完成创建分组任务、更改默认通知消息等高级任务。

2.5.8 复制中心

通过以下方式可以启动“复制中心”，窗口如图 2.13 所示。

- ☑ 选择“开始”→“所有程序”→IBM DB2→“DB2COPY1（默认值）”→“一般管理工具”→“复制中心”命令。
- ☑ 按下 Windows+R 快捷键，在弹出的“运行”对话框中输入 db2rc，然后单击“确定”按钮。
- ☑ 在“控制中心”、“配置助手”、“命令编辑器”、“任务中心”、“运行状况中心”或“日志”窗口的主菜单中选择“工具”菜单项，在列出的子菜单中选择“复制中心”菜单项。
- ☑ 在“控制中心”、“配置助手”、“命令编辑器”、“任务中心”、“运行状况中心”或“日志”窗口的工具栏中单击按钮。

“复制中心”通过定义将指定的更改从一个位置复制到另一个位置，使两处数据同步。“复制中心”和启动板将引导用户完成一些基本复制功能。可以在启动板与“复制中心”对象视图之间替换执行复制任务的方式。

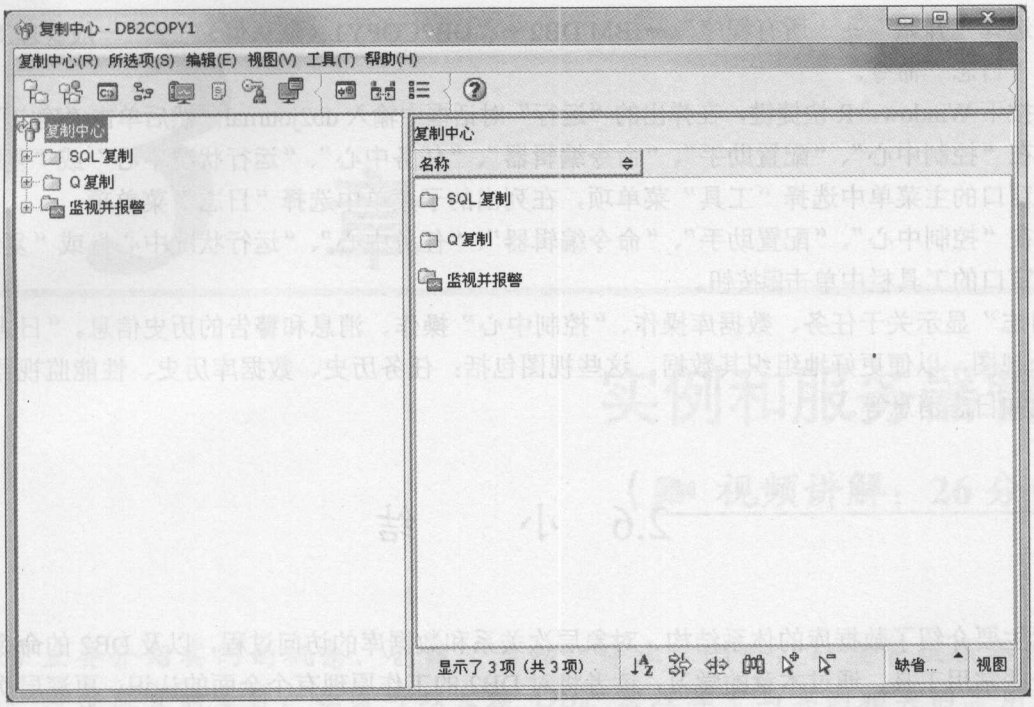


图 2.13　“复制中心”窗口

2.5.9　日志

通过以下方式可以启动“日志”窗口，窗口如图 2.14 所示。

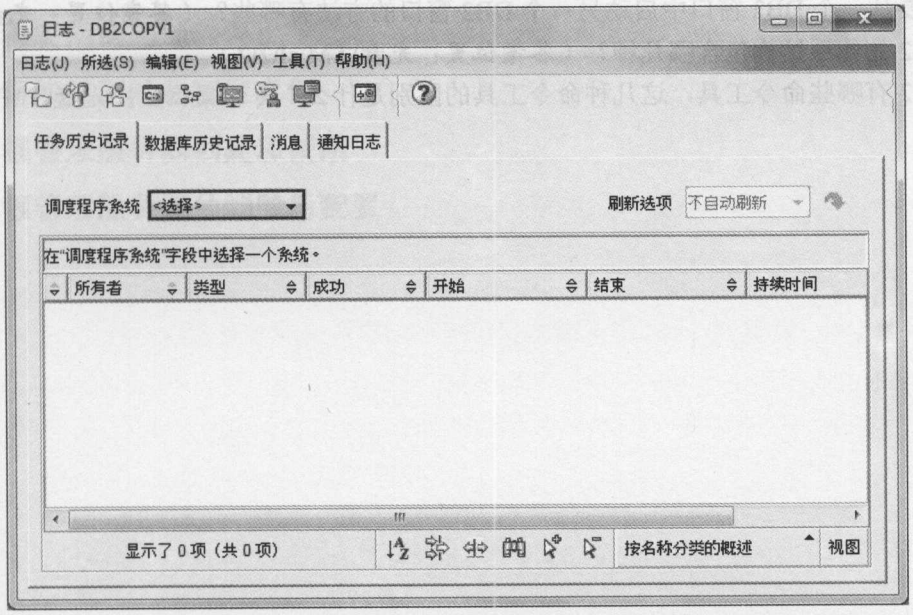


图 2.14　“日志”窗口

- ☑ 选择“开始”→“所有程序”→IBM DB2→“DB2COPY1（默认值）”→“一般管理工具”→“日志”命令。
- ☑ 按下 Windows+R 快捷键，在弹出的“运行”对话框中输入 db2journal，然后单击“确定”按钮。
- ☑ 在“控制中心”、“配置助手”、“命令编辑器”、“任务中心”、“运行状况中心”或“复制中心”窗口的主菜单中选择“工具”菜单项，在列出的子菜单中选择“日志”菜单项。
- ☑ 在“控制中心”、“配置助手”、“命令编辑器”、“任务中心”、“运行状况中心”或“复制中心”窗口的工具栏中单击按钮。

“日志”显示关于任务、数据库操作、“控制中心”操作、消息和警告的历史信息。“日志”准备了若干个视图，以便更好地组织其数据。这些视图包括：任务历史、数据库历史、性能监视器警告、消息、通知日志消息等。

2.6 小　　结

本章主要介绍了数据库的体系结构、对象层次关系和数据库的访问过程，以及 DB2 的命令工具和其他的一些常用工具。通过本章的学习，读者将对 DB2 的工作原理有个全面的认识，更深层次的内容我们将在本书后续章节中继续讨论。

2.7 实践与练习

（1）想要从一个 DB2 窗口中启动另一个 DB2 窗口的方法有哪些？（**答案位置：光盘\TM\sl\2\1**）

（2）DB2 的体系结构包含哪几种？（**答案位置：光盘\TM\sl\2\2**）

（3）DB2 有哪些命令工具，这几种命令工具的区别是什么？（**答案位置：光盘\TM\sl\2\3**）

第 3 章

实例和服务器管理

（视频讲解：26 分钟）

本章主要介绍实例的概念、创建、启动、停止、删除，以及管理服务器 DAS 等内容。通过本章的学习，读者可以掌握 DB2 数据库中与实例相关的常用命令，并掌握管理服务器 DAS 的概念。

通过阅读本章，您可以：

- 了解 DB2 实例的基本概念
- 掌握实例的创建和使用
- 掌握实例的启动和停止
- 掌握如何删除和配置实例
- 掌握管理服务器创建和使用
- 掌握管理服务器的删除与配置

3.1 实　　例

实例用来为数据库提供安全、通信、内存分配和进程间通信等功能，数据库一定要在实例下创建，这样一来，数据库只负责前台正常的运行，而一些后台的事情则由实例来进行管理。

3.1.1 什么是 DB2 实例

DB2 的实例是一个 DB2 服务，DB2 实例创建后产生一个用于运行 DB2 的系统服务。在这里，可以把实例想象成是数据库的一个运行环境，数据库一定要在一个实例下创建。在一个系统中，用户可以创建若干个实例，每一个实例使用各自不同的端口服务于远程应用程序。每一个实例可以包含若干个数据库，每个数据库使用名字来区分。换句话说，连接到一个端口，就是连接到实例。系统、实例和数据的关系如图 3.1 所示。

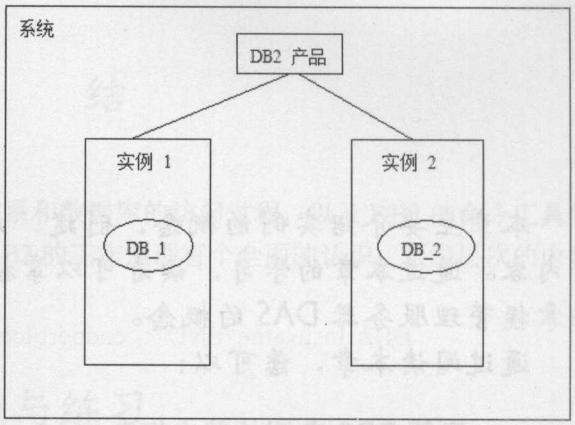


图 3.1　系统、实例和数据的关系

如果一个实例停止了，那么该实例下的所有数据库将不可用。一个实例的停止不会影响到其他实例下的数据库。

3.1.2 创建实例

在 Windows 系统下，实例不需要与用户绑定，在不同实例间切换时，通过 db2instance 环境变量进行设置。

在 Windows 平台下，安装 DB2 软件时会自动创建一个名为 DB2 的实例，用户可以使用这个实例，也可以自己创建实例。通过 db2cmd 进入命令窗口执行创建实例的命令 db2icrt，语法如下：

```
db2icrt 使用程序创建数据库实例:
db2icrt   InstName [-s   {ese | wes | standalone | client }]
                   [-p   instance profile path]
                   [-u   username,password]
                   [-h   hostname]
                   [-r   baseport,endport]
                   [-j   textSearchConfig]
                   [-? ]
```

参数说明：

☑ -s 选项用于指定实例的类型。Ese 参数用于创建具有分区数据库支持的 DB2 数据库服务器的一个实例，该数据库服务器带有本地和远程客户机。wse 参数用于创建 DB2 数据库服务器的一个实例，该数据库服务器带有本地和远程客户机。此类型是 DB2 工作组版、DB2 易捷版或 Express-C 版以及 DB2 Connect 企业版的默认实例类型。standalone 参数用于创建 DB2 数据库服务器的一个实例，该数据库服务器带有本地客户机。此类型是 DB2 个人版的默认实例类型。client 参数用于创建 IBM 数据服务器客户机的一个实例。此类型是 IBM 数据服务器客户机产品和 DB2 Connect 个人版的默认实例类型。

☑ -p 用于指定实例概要文件路径。

☑ -u 用于指定 DB2 服务的账户名和密码。创建 ese 实例时需要此选项。

☑ -h 用于覆盖默认 TCP/IP 主机名（如果当前机器有多个 TCP/IP 主机名）。在创建默认节点（节点 0）时，将使用该 TCP/IP 主机名。

☑ -r 用于指定在 MPP 方式下运行时，分区数据库实例要使用的一系列 TCP/IP 端口。如果指定了此选项，那么本地机器的 services 文件将使用下列条目进行更新：

```
DB2_InstName          baseport/tcp
DB2_InstName_1        baseport+1/tcp
DB2_InstName_2        baseport+2/tcp
DB2_InstName_END      endport/tcp
```

☑ -j 不能与 client 实例一起使用。textSearchConfig 的值是“TEXT_SEARCH”、“TEXT_ SEARCH，servicename”、“TEXT_SEARCH，servicename，portnumber”或“TEXT_SEARCH，portnumber”。将使用提供的服务名称和 TCP/IP 端口号（如果指定）配置 DB2 文本搜索实例服务，否则将生成默认值。

☑ -?用于显示帮助。

3.1.3　实例目录

实例创建后，会生成实例目录，实例目录存储着与数据库实例相关的所有信息。实例目录一旦创建，就不能更改其路径。

实例目录包含：

（1）数据库管理器配置文件（db2systm）。

（2）系统数据库目录（SQLDBDIR）。

（3）节点目录（SQLNODIR）。

（4）节点配置文件（db2nodes.cfg），db2nodes.cfg 文件用来定义参与 DB2 实例的数据库分区服务器。如果想要将高速互联用于数据库分区服务器通信，那么还可以使用 db2nodes.cfg 文件来指定高速互联的 IP 地址或主机名。

（5）诊断文件、数据库错误日志等。

3.1.4 列出实例

在实例创建后，可以用相关的命令来管理实例。在使用实例之前，必须更新每个用户的数据库环境，以便该环境可以访问实例并运行 DB2 实例相关命令。实例相关的命令对权限要求很高，例如 db2icrt 和 db2idrop 命令需要 root 权限才能执行。除了这两个命令以外的命令需要具有 SYSADM、SYSCTRL、SYSMAINT 的权限才能运行。

通过 db2cmd 进入命令窗口，执行列出机器上 DB2 实例的命令：db2ilist，代码如下所示。

```
C:\Users\mr>db2ilist
TTS
TESTINST
DB2
```

3.1.5 启动/停止实例

实例创建后，需要通过 db2start 命令启动才能工作。在 Windows 平台下，通过 db2cmd 进入命令窗口，执行 db2start。如果存在多个实例，则需要先切换到相应实例后才能启动，切换的命令是通过 db2instance 环境变量实现的。代码如下所示。

```
C:\Users\mr>set db2instance=testinst          # 设置实例
C:\Users\mr>db2 get instance                  # 查看当前实例
    当前数据库管理器实例是：DB2
C:\Users\mr>db2start                          # 启动实例
SQL1026N  数据库管理器已活动。
```

停止实例的命令是 db2stop。

```
C:\Users\mr\db2stop
SQL1064N  DB2STOP 处理成功。
```

3.1.6 更新实例

通过 db2cmd 进入命令窗口，执行实例更新命令 db2iupdt，一般在打补丁或版本升级时使用。db2iupdt 命令需要 root 用户执行，执行前需要首先停止实例。db2iupdt 使用方法如下：

```
db2iupdt  InstName     # InstName 是实例名称
      /u:username,password
      [/p:instance  profile  path]
      [/r:baseport,endport]
      [/h:hostname]
```

```
[/?]
[/q]
[/a:authType]
[/j  textSearchConfig]
```

InstName 实例名

/u　　用于指定 DB2 服务的账户名和密码。当创建分区数据库实例时，此选项是必需的

/p　　用于指定已更新实例的新的实例概要文件路径

/r　　用于指定当前 MPP 方式下运行时，分区数据库实例要使用的一系列 TCP/IP 端口。

如果指定此选项，那么本地机器的 services 文件将使用下列条目进行更新：

DB2_InstName　　baseport/tcp

DB2_InstName_END　endport/tcp

/h　　用于覆盖默认 TCP/IP 主机名

/?　　此用法信息

/q　　安静方式

/a　　authType 是实例的认证类型（SERVER、CLIENT 或 SERVER_ENCRYPT）

示例：如果在创建实例后安装了 DB2 工作组服务器版或 DB2 企业服务器版，可输入一下命令来更新实例。



注意

db2iupdt 通常是给小版本打补丁，例如 DB2 V9.7.0.1 升级到 V9.7.0.3 用 db2iupdt 来更新。

3.1.7　删除实例

通过 db2cmd 进入命令窗口执行删除实例的命令 db2idrop，该命令要求使用 root 用户执行，删除前必须停止实例。

```
C:\Users\mr>db2idrop InstName      # InstName 是实例的名称
DB20000I  DB2IDROP  命令成功完成。
```



注意

删除实例并不会删除该实例下的数据库。

3.1.8　配置实例

每个实例都有实例配置文件（db2systm），实例配置文件用来控制实例的安全、通信、管理和资源的分配。可以根据需要查看、更改和复位这个配置参数。这个配置文件是二进制的，只能通过命令来修改。

通过 db2cmd 进入命令窗口，执行查看当前实例配置参数的命令是：

```
db2 get dbm cfg
```

此时会显示安装该 DB2 产品期间指定的默认参数值，或在先前更新配置参数期间指定的那些值。如果想要将配置参数复位为建议的默认值，请输入：

```
db2 reset dbm cfg
```

3.2 管理服务器

当远程 DB2 客户端需要通过图形界面对数据库进行定时备份时，就需要启动数据库管理服务器了。

3.2.1 管理服务器的概念

DAS 是数据库管理服务器（Database Administration Server）的缩写，它是特殊的 DB2 管理控制点，用于帮助其他 DB2 服务器执行远程管理任务（其他远程 DB2 服务器需要安装 Database Administration Client，DB2 V9 以后更名为 DB2 Client）。DAS 响应来自远程 DB2 服务器管理工具和配置助手的请求。如果要使用“客户机配置助手”来发现远程数据库或随 DB2 产品提供的图形工具，那么 DAS 必须正在运行。

DAS 与实例是一对多的关系，即 DB2 DAS 中只能有一个 DAS，但这个 DAS 可以同时管理多个实例。DAS 允许使用 DB2 工具对服务器上的数据库进行本地和远程管理。DB2 图形管理工具与 DAS 共同执行以下任务：

（1）从远程客户机上管理、调度 DB2 或操作系统脚本以在服务器上运行。

（2）使用 DB2 Discovery 自动设置客户机/服务器连接。

（3）启动停止远程 DB2 实例。

3.2.2 创建管理服务器

管理服务器必须位于将要被管理和搜索的每台服务器上。每个服务器只能创建一个 DAS 实例。除非另外指定，否则，这个实例会在 DB2 安装过程中自动建立。在 Windows 操作系统中，DAS 的实例名是 DB2DAS00，如图 3.2 所示。

在 Windows 中，创建 DAS 时必须具有本地 Administrator 权限。如果使用特定用户账户，那么在发出 db2admin create 命令时必须使用“/USER:”和“/PASSWORD:”。

在数据库服务器中只能有一个 DAS。如果已经创建了 DAS，那么需要用 db2admin drop 命令将其删除。

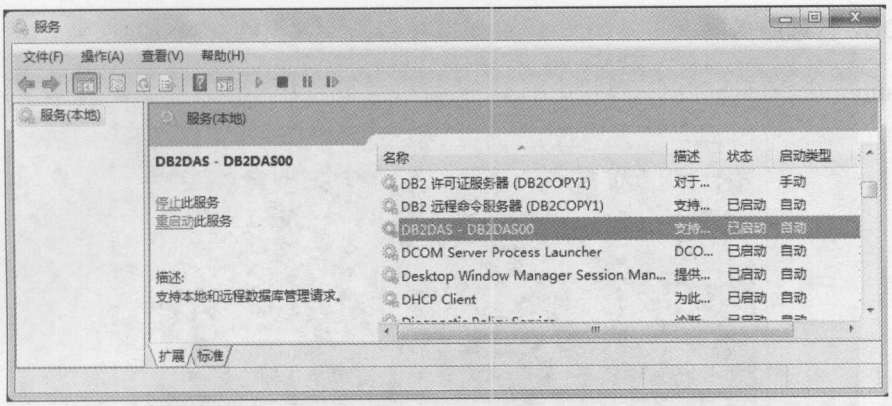


图 3.2　在 Windows 中默认创建的 DAS

创建 DAS 后，使用 db2admin 命令设置或更改登录账户，代码如下：

```
db2admin setid <userName> <passWord>
```

<userName>和<passWord>是具有本地管理员权限的账户的用户名和密码。在运行此命令前，必须使用具有本地管理员权限的账户或用户标识登录计算机。

注意

在 Windows 中，不要使用控制面板中的服务来更改 DAS 的登录账户，否则不能为登录账户设置某些访问权限。

当创建 DAS 时，可以选择是否提供账户名和密码。如果账户名和密码有效，它们将标识 DAS 的所有者。不要使用为 DAS 创建的用户标识或账户名作为“用户账户”。将账户密码设置为“密码永不到期”。在创建 DAS 之后，就可以使用 db2admin setid 命令提供账户名和密码。

3.2.3　管理服务器的相关命令

1. 启动和停止 DB2 管理服务器

在 Windows 系统中，通过手动的方式启动或停止 DAS，在启动或停止前必须使用 Administrators、Server Operators 或 Power Users 组的账户或用户标识登录计算机。

要启动 DAS 可使用 db2admin start 命令，如图 3.3 所示。

要停止 DAS 可使用 db2admin stop 命令，如图 3.4 所示。

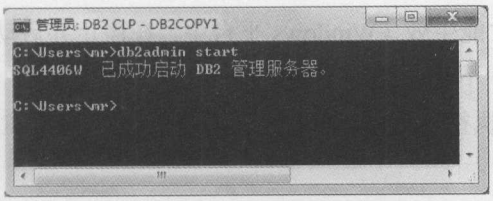


图 3.3　启动 DAS

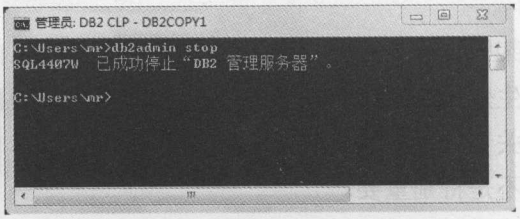


图 3.4　停止 DAS

2. 列出 DB2 管理服务器

使用 db2admin 命令可以列出计算机中 DAS 的名称，如图 3.5 所示。

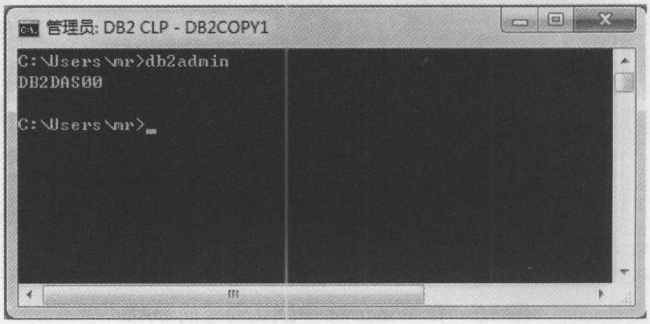


图 3.5　列出 DAS 名称

3.2.4　删除管理服务器

要删除 DAS，需要在 Windows 系统中确保有正确的登录权限，然后使用 db2admin drop 命令来删除 DAS，删除命令如下：

```
db2admin drop
```

注意

从 DB2 V9.7 版本开始，DAS 将逐渐退出历史舞台。

3.2.5　配置管理服务器

通过 db2cmd 进入命令窗口，查看 DAS 的当前配置参数，命令如下：

```
db2 get admin cfg
```

使用此命令会显示默认配置参数的当前值，或是更新配置参数期间指定的那些值，如图 3.6 所示。

将配置参数恢复成默认值，命令如下：

```
db2 reset admin cfg
```

某些情况下，将 DAS 配置文件的更改装入内存后才能生效（Windows 平台中，在停止并启动该服务时生效）。在其他情况下，配置参数是可联机配置的，不必重新启动 DAS 更改就可以生效。

关于 DAS 这个概念，大家了解下就可以，它主要执行远程图形界面管理功能，这个概念在 DB2 中并不是特别重要。

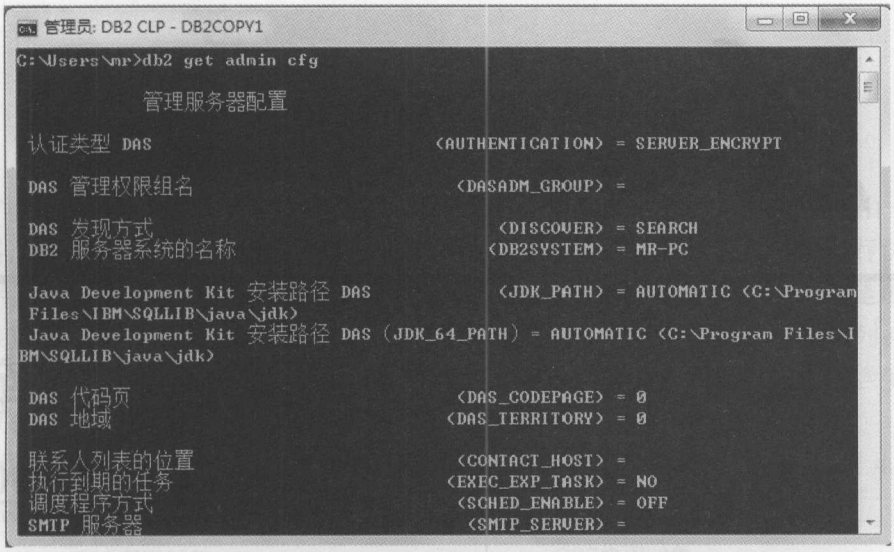


图 3.6　当前配置参数

3.3　小　　结

本章向读者重点讲解实例的概念，实例、系统与数据之间的关系以及实例的常用命令，对于 DAS 管理服务器，大家有所了解即可。

3.4　实践与练习

（1）什么是 DB2 实例？（**答案位置：光盘\TM\sl\3\1**）

（2）如何创建实例？（**答案位置：光盘\TM\sl\3\2**）

（3）分别列举启动、列出、删除和更新实例的命令。（**答案位置：光盘\TM\sl\3\3**）

第 4 章

DB2 数据库基础

（视频讲解：30 分钟）

本章主要讲解如何使用“控制中心”和“命令编辑器”对数据库进行创建、修改和删除等操作，通过本章的学习，读者可以熟悉 DB2 数据库的组成元素，并能够掌握创建和管理数据库的方法。

通过阅读本章，您可以：

- 掌握数据库的基本结构
- 掌握数据库的基本概念
- 掌握数据库的创建、编目和删除
- 熟悉如何在 SQL 语句中添加注释

4.1　数据库结构

一个实例可以包含多个数据库，但一个数据库只能属于一个实例。每个数据库都是由一组对象组成的，如表、视图、索引、触发器和存储过程等，以便随时对数据库中的数据及其对象进行访问和管理。

DB2 是利用逻辑存储模型和物理存储模型来处理数据的，DB2 的逻辑层次关系是由实例→数据库→表空间→表组成。用户操作的实际数据存放在表中。表是二维结构，由一些行和列构成，表本身存放在表空间中，表空间是存放表的容器，也是数据库的逻辑存储层。每个数据库包含多个表空间，一个表空间可以包含多个表，每个表空间只能属于一个数据库。

根据数据库的管理方式，表空间分为系统管理（SMS）和数据库管理（DMS）。图 4.1 是 DB2 数据库对象逻辑结构图。

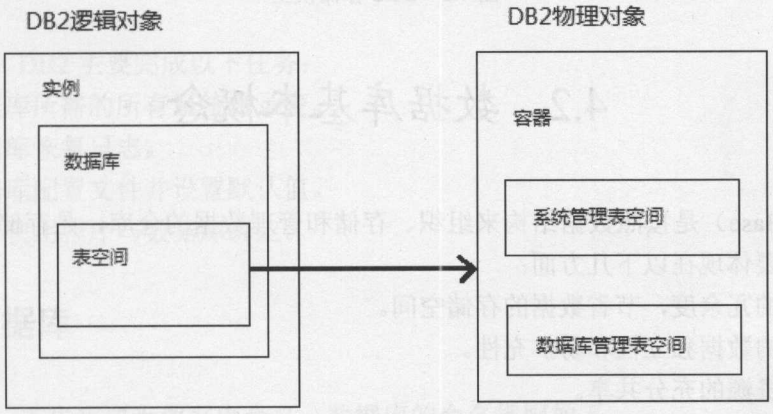


图 4.1　数据库对象逻辑层次关系

对于初学者来说，不需要关心表数据在内部是如何存储的，只要关心表的逻辑结构和表间关系。但是对于 DBA 来说，为了更好地规划存储设计和优化 I/O 性能，有必要深入研究 DB2 的内部存储结构。

在操作系统中最小的存储单位是块（block），在 Windows 系统中，最小的存储单位为 1KB。而数据库中最小的存储单位是数据页（datapage）。DB2 将表和索引数据存在页里，页是 DB2 的最小物理分配单元，表中每行数据只能包含在一页，不能跨页。DB2 数据库中有：4KB、8KB、16KB 和 32KB 几种数据页。可以根据业务类型和表的大小来选择合适的数据页。假如一行数据大小为 25KB，那么需要的页的大小应该是 32KB。

DB2 在读取数据时，为了保证写的吞吐量，并不会每次读取一页，而引入了更大的单位 EXTEND（一组连续的页），它的大小是 datapage 的整数倍。当一个表空间有多个容器，为了数据的均衡分布，DB2 在写数据时，会按照循环的方式在各个容器里写数据，即在一个容器写满一个 EXTEND 后，开始第二个容器继续写 EXTEND，如此循环，以确保数据均匀分布在多个容器上，从而提高读写效率。

每个表空间由一个或多个容器构成，表空间仅仅是逻辑存储层，具体的数据是存在容器里。容器是由多个 EXTEND 组成的。

关于表空间、容器、EXTEND、PAGE 之间的关系如图 4.2 所示，它们构成了 DB2 的存储模型。

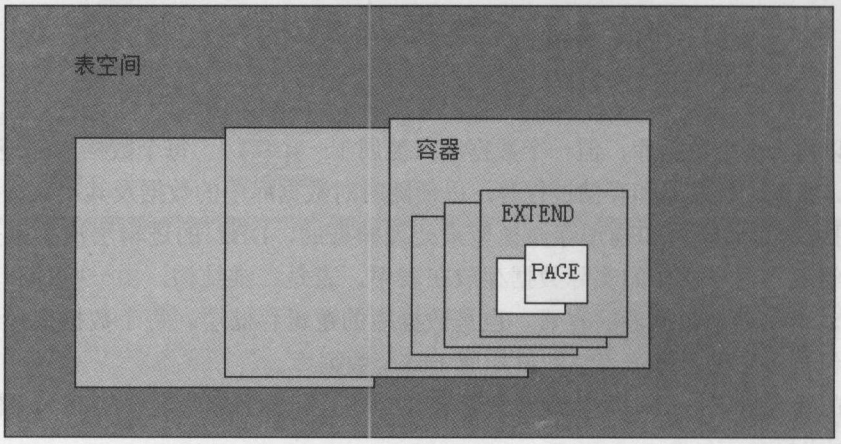


图 4.2　DB2 存储模型

4.2　数据库基本概念

数据库（DataBase）是按照数据结构来组织、存储和管理数据的仓库，是存储在一起的相关数据的集合，其优点主要体现在以下几方面：

☑ 减少数据的冗余度，节省数据的存储空间。

☑ 具有较高的数据独立性和易扩充性。

☑ 实现数据资源的充分共享。

下面介绍与数据库相关的几个概念。

（1）数据库系统。数据库系统（DataBase System，DBS）是采用数据库技术的计算机系统，是由数据库（数据）、数据库管理系统（软件）、数据库管理员（人员）、硬件平台（硬件）和软件平台（软件）5 部分构成的运行实体。其中数据库管理员（DataBase Administrator，DBA）是对数据库进行规划、设计、维护和监视等的专业管理人员，在数据库系统中起着非常重要的作用。

（2）数据库管理系统。数据库管理系统（DataBase Management System，DBMS）是数据库系统的一个重要组成部门，是位于用户与操作之间的一层数据管理软件，负责数据库中的数据组织、数据操纵、数据维护和数据服务等。主要具有如下功能。

☑ 数据存取的物理构建：为数据模式的物理存取与构建提供有效的存取方法与手段。

☑ 数据操纵功能：为用户使用数据库的数据提供方便，如查询、插入、修改、删除等以及简单的算术运算和统计。

☑ 数据定义功能：用户可以通过数据库管理系统提供的数据定义语言（Data Definition Language，DDL）方便地对数据库中的对象进行定义。

☑ 数据库的运行管理：数据库管理系统统一管理数据库的运行和维护，以保障数据的安全性、完整性、并发性和故障的系统恢复性。

☑ 数据库的建立和维护功能：数据库管理系统能够完成初始数据的输入和转换、数据库的转储和恢复、数据库的性能监视和分析等任务。

（3）关系数据库。关系数据库是支持关系模型的数据库。关系模型由关系数据结构、关系操作集合和完整性约束 3 部分组成。

☑ 关系数据结构：在关系模型中数据结构单一，现实世界的实体以及实体间的联系均用关系来表示，实际上关系模型中数据结构就是一张二维表。

☑ 关系操作集合：关系操作分为关系代数、关系演算、具有关系代数和关系演算双重特点的语言（SQL 语言）。

☑ 完整性约束：完整性约束包括实体完整性、参照完整性和用户定义的完整性。

4.3　数据库的创建、编目和删除

创建数据库时，DB2 主要完成以下任务：

（1）设置数据库所需的所有系统目录表。

（2）分配数据库恢复日志。

（3）创建数据库配置文件并设置默认值。

（4）将数据库使用程序与数据库绑定。

4.3.1　创建数据库

要创建数据库，首先需要为数据库命名，数据库的命名规则如下。

☑ 数据库的名称可以由以下字符组成：a～z、A～Z、0～9、@、#和$。

☑ 名称中的第一个字符必须是字母表字符、@、#或$，不能是数字或字母序列 SYS、DBM 或 IBM。注意，数据库的名称不能超过 8 个字母。

☑ 数据库名称或数据库别名是唯一的字符串，包含前面扫描的 1 到 8 个字母、数字或键盘字符。

创建 DB2 数据库的方法主要有两种，分别是：使用界面方式和使用命令方式，下面分别进行讲解。

1. 使用界面方式创建数据库

创建数据库的步骤如下：

（1）打开“控制中心”窗口，在对象视图中选择实例 DB2 下的“数据库”文件夹，在该文件夹的右键快捷菜单中选择“创建数据库”→“标准”命令，如图 4.3 所示，从而启动“创建数据库向导”对话框。

（2）在“创建数据库向导”对话框的“指定新数据库的名称”界面中，在“数据库名称”文本框中输入“MRKJ”。在“默认路径”文本框中输入“E:\”。在“别名”文本框中输入“MRKJ”，如果别名为空，会默认使用数据库名称作为别名。在“注释”文本框中输入“学生成绩数据库”。默认的管理存储器为“让 DB2 管理存储器（自动存储器）”。默认缓冲池和表空间页大小为“4KB”，如图 4.4 所示。

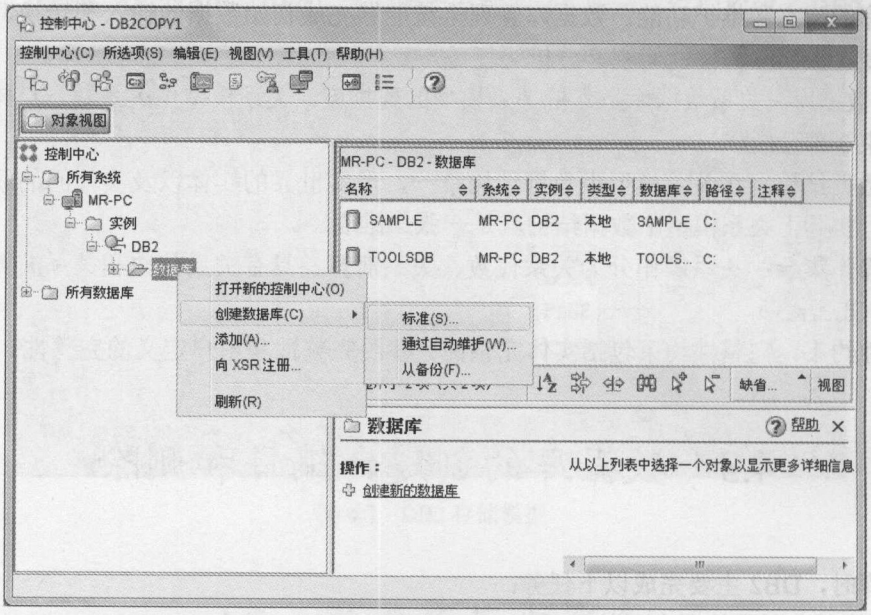


图 4.3 “创建数据库”菜单

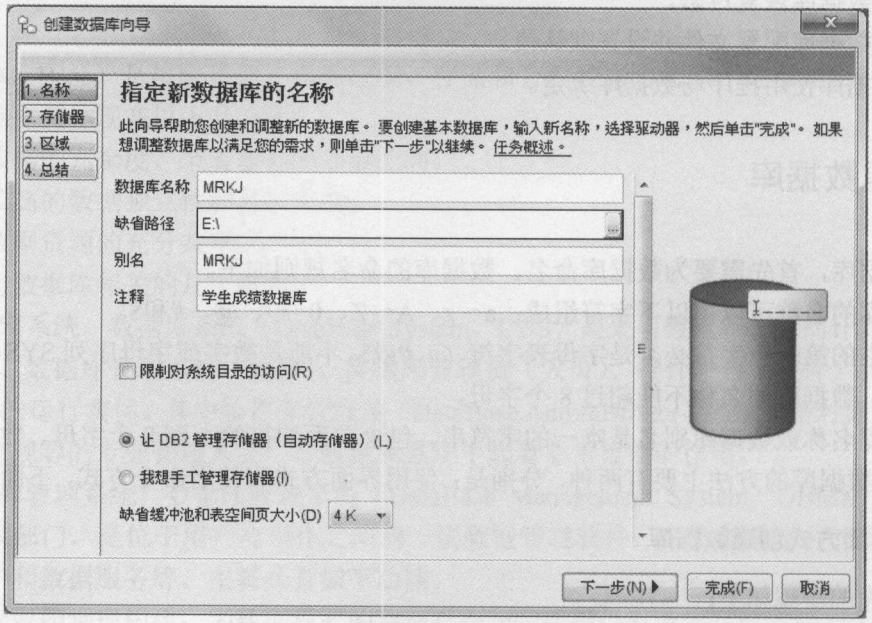


图 4.4 “创建数据库向导”对话框

注意

数据库不能用中文来命名，否则无法创建数据库。

（3）在“创建数据库向导”对话框的“指定用来存储数据的位置”界面中，如果选中“使用数据库路径作为存储路径：E:\”复选框，系统会将第（2）步中指定的数据库路径用作 MRKJ 数据库的存

储路径。MRKJ 数据库将建在数据库默认路径“E:\”下，如果单击“添加”按钮指定其他存储路径，则数据库会建在其他路径上。如图 4.5 所示，单击“下一步”按钮继续。

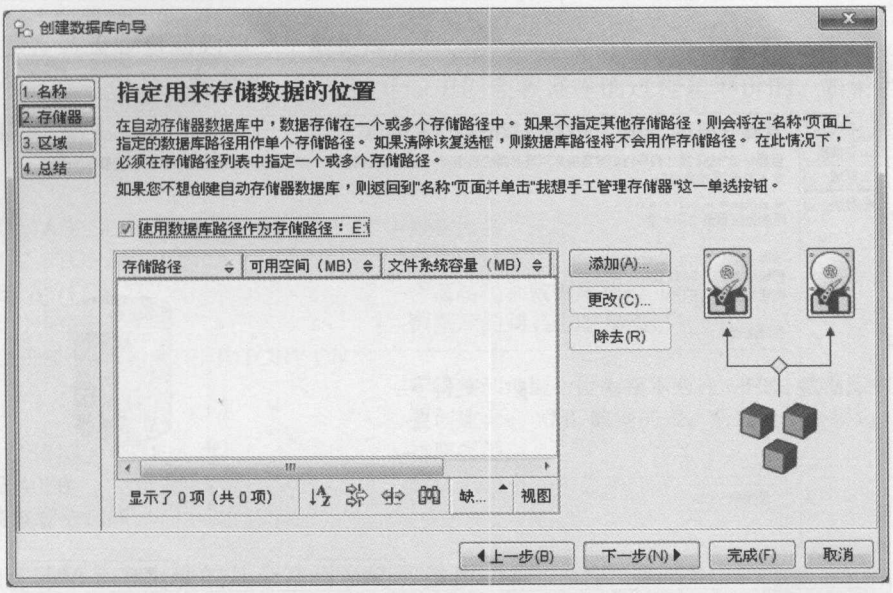


图 4.5　指定用来存储数据的位置

（4）在“创建数据库向导”对话框的“指定此数据库的语言环境”界面中，在“国家/地区”下拉列表框中选择“中华人民共和国”，默认值为创建数据库的客户机系统语言环境，因为这里系统语言环境也是“中华人民共和国”，所以这个列表框中选择默认选项也可以。在“代码集”下拉列表框中必须选择“UTF-8”选项（因为 MRKJ 数据库中将包含 XML 列，必须在使用 UTF-8 代码集兼容的地域代码定义的数据库中创建具有 XML 列的表）。在“整理顺序”栏中，默认选中“系统”单选按钮，即根据选择的代码集来排序，如图 4.6 所示。单击“下一步”按钮。

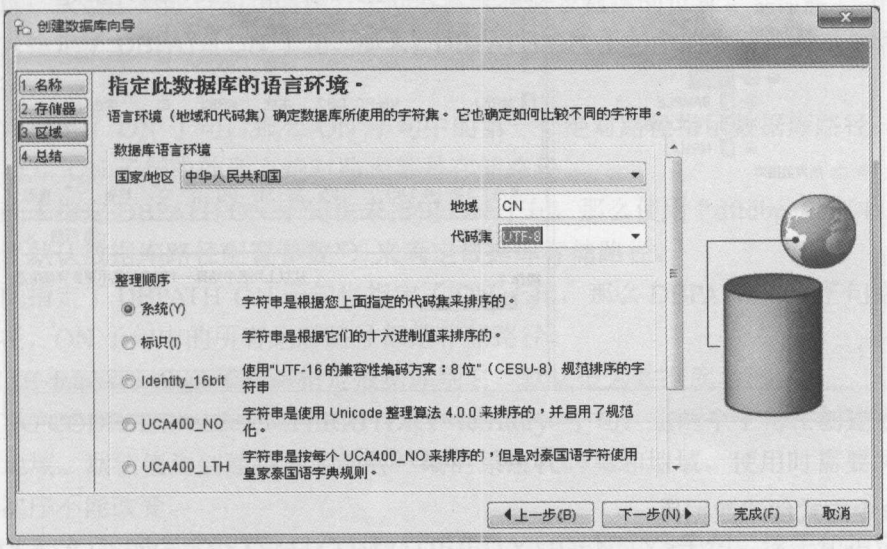


图 4.6　指定此数据库语言环境

（5）在“创建数据库向导”对话框的“复查在单击‘完成’之后将执行的操作”界面中，如果单击“显示命令”按钮，可以查看自动生成的 SQL 语句，检查无误后可以单击“完成”按钮，如图 4.7 所示。

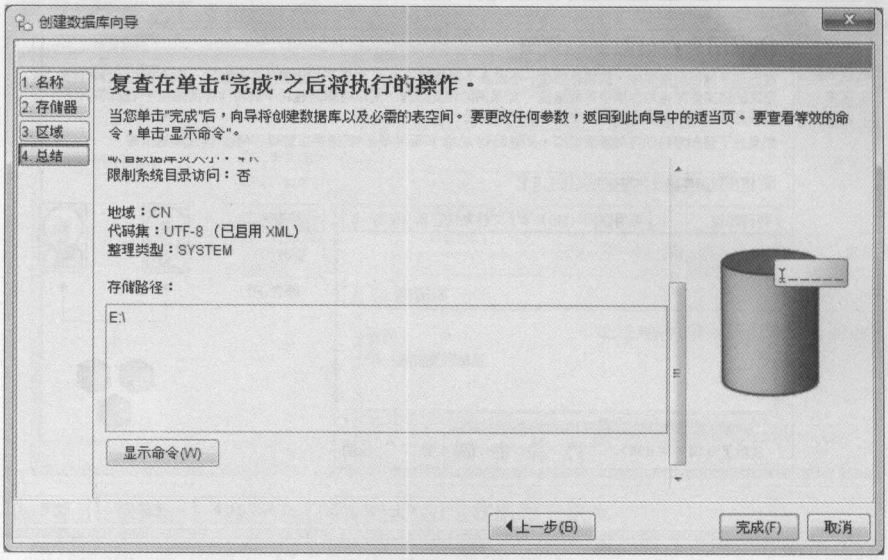


图 4.7　复查操作内容

（6）创建成功，在“控制中心”的对象视图中就可以看到 MRKJ 数据库了，如图 4.8 所示。

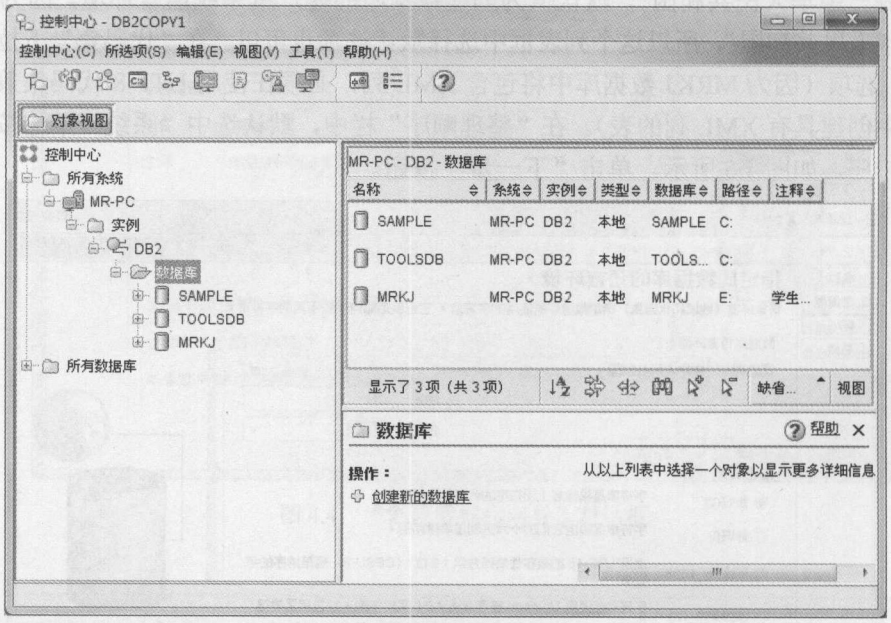


图 4.8　MRKJ 数据库

2．使用命令方式创建数据库

虽然创建数据库命令的语法相当复杂，但是在实际使用“命令编辑器”时就不用这么复杂了。可

以在“命令编辑器”输入下面的语句然后单击 ▶ 按钮。

```
CREATE DATABASE database_name;
```

系统会按照默认设置自动创建一个数据库。

下面的命令 DBA 会十分感兴趣，它与在界面中创建数据库的过程是等价的。如果当前不存在或者已经删除了 MRKJ 数据库，则可以在“命令编辑器”中输入下面的命令，然后单击 ▶ 按钮，创建 MRKJ 数据库。

```
CREATE DATABASE database_name          --数据库名
AUTOMATIC STORAGE YES                  --自动存储器
ON 'E:\' DBPATH ON 'E:\'               --存储路径和数据库路径
ALIAS MRKJ                             --数据库的别名也叫 MRKJ
USING CODESET UTF-8 TERRITORY CN
                                       --字符集和地域，因为接下来在 MRKJ 数据库中
                                       --要创建含有 XML 数据的表，所以此处必须为 UTF-8 编码格式
COLLATE USING SYSTEM                   --排序类型
PAGESIZE 4096                          --默认页大小
WITH '学生成绩数据库'
```

下面详细解释命令中出现的几个常用子句和参数。

（1）database_name：要创建的数据库名称。

（2）AUTOMATIC STORAGE <YES | NO>子句：指定数据库存储器方式。默认情况下，数据库管理器会将所有数据库都作为“自动存储器”数据库来创建。要创建“手工管理存储器”数据库时，参数选择 NO。

（3）[ON {path} [,…n]] [DBPATH ON driver]子句：path 是绝对路径，可以是路径或盘符，最大路径长度是 175 个字符。[,…n]表示可以有多个路径或盘符，它们之间要用逗号隔开。driver 必须是盘符。

数据库路径是：数据库管理器存储数据库的各种控制文件的位置。

存储器路径是：数据库创建自动存储器表空间容器来存储表数据的位置。

- ☑ 如果只指定了 DBPATH ON 子句，那么，DBPATH ON 子句中的盘符既是数据库路径，又是存储路径。
- ☑ 如果只指定了 ON 子句，那么 ON 子句中的第一个绝对路径指示数据库路径（而不仅仅是存储路径），ON 子句中的所有绝对路径都是存储路径。
- ☑ 如果既未指定 DBPATH ON 子句也未指定 ON 子句，那么使用“dftdbpath”（也就是 DB2 系统中的“默认数据库路径配置参数”）来确定数据库存储路径。
- ☑ 如果既指定了 DBPATH ON 子句也指定了 ON 子句，那么 DBPATH ON 子句中的盘符是数据库路径，ON 子句中的所有绝对路径都是存储路径。

（4）ALIAS database_alias 子句：指定数据库别名，默认值为数据库名称。

（5）USING CODESET<codeset>TERRITORY<territory>子句：这两个子句在创建数据库时指定想要的代码集和地域。默认值为创建数据库的客户端的系统代码集和地域。使用时需要两个子句一起使用，而且先后次序不能改变。

（6）COLLATE USING<SYSTEM | COMPATIBILITY | IDENITY>子句，该子句指定数据库的字符

串整理排序类型。它的默认参数为 SYSTEM。参数 SYSTEM 指字符串是以系统的代码集来排序的。参数 COMPATIBILITY 是指字符串使用 DB2 版本 2 整理顺序来排序，DB2 版本 2 为排序做了专门的优化。在创建需要使用 DB2 版本 2 整理顺序来排序字符串的数据库时，请选择此项。一旦数据库创建完成后，该属性不能被修改。

（7）PAGESIZE<integer>子句：指定数据库默认的表空间的页大小。

（8）WITH “comment-string” 子句：指定用来描述数据库的注释。



注意

可以在一个主机上创建相同名称的数据库，但不能创建别名相同的数据库。

在发出 CREATE DATABASE 命令时，DB2 会创建许多文件。这些文件包含日志文件、配置信息、历史文件和 3 个默认的表空间。这些表空间如下。

- ☑ SYSCATSPACE：这是保存 DB2 系统编目的地方，系统编目跟踪与 DB2 对象相关联的所有元数据，即通常所说的“数据字典”。
- ☑ TEMPSPACE1：DB2 用来放置分组、排序、连接和重组中间结果的临时工作区域。
- ☑ USERSPACE1：默认情况下存放所有用户对象（表、索引）的地方。

4.3.2 SQL 语句中的注释符

在 SQL 语句中有两类注释符，分别如下。

（1）ANSI 标准的注释符 “--”，用于单行注释。例如：

```
create database uus;        --创建 uus 数据库
```

（2）注释符 “/*” 和 “*/” 配对使用。可用于多行注释。其中，“/*” 用于注释文字的开头，“*/” 用于注释文字的结尾。例如：

```
/*  创建一个数据库，
    数据库名字叫 uus */
```

4.3.3 编目数据库

编目数据库的操作包括：除去数据库、添加数据库和更改数据库。更改数据库其实就是先除去数据库，然后再添加刚刚被除去的数据库，并修改该数据库的某些属性。

1. 用界面方式除去数据库

“除去” 的含义是：将被除去的数据库可以是本地数据库（即本机上的数据库），也可以是网络数据库（即其他主机上的数据库）。如果它是本地数据库，则它必须已经存在于本地，如果它是其他主机（服务器）上的数据库，则它必须已经被添加到本地。也就是说，该数据库已经显示在 “控制中心”

窗口的对象视图中（同时也会显示在“配置助手”窗口的列表中）。除去数据库后，该数据库就不能显示在“控制中心”窗口的对象视图中（也不能显示在“配置助手”窗口的列表中）。除去该数据库之后，不可以连接、查询或更改该数据库；但该数据库并没有被真正被删除，它仍然存在于被除去之前的位置。

可以通过“配置助手”或“控制中心”两种方式中的一种除去数据库。

用“配置助手”除去数据库的方法如下：

（1）在“配置助手”窗口中选择将要除去的数据库，例如 MRKJ，然后单击鼠标右键，在弹出的快捷菜单中选择“除去数据库”命令，如图 4.9 所示。

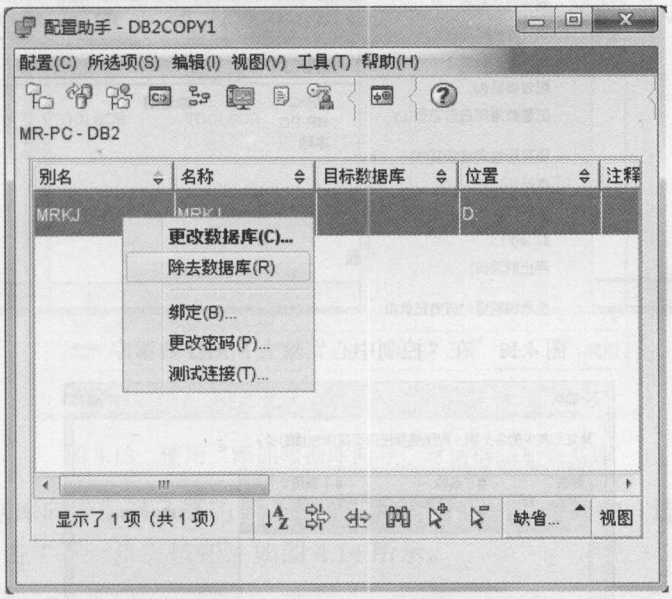


图 4.9　使用“配置助手”除去数据库

（2）在弹出的“确认”对话框中，检查即将被除去的数据库名称是否正确，如果正确无误，则单击“确定”按钮除去该数据库，如图 4.10 所示。

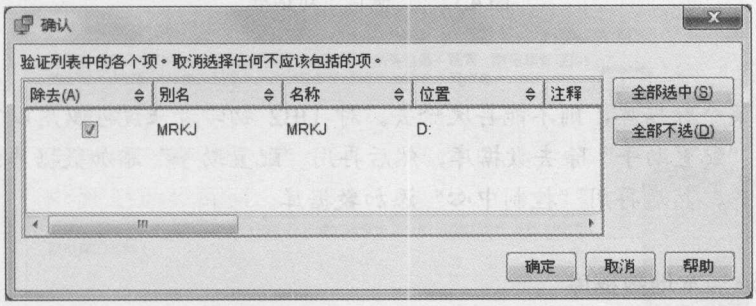


图 4.10　确认除去数据库

用“控制中心”除去数据库的方法如下：

在“控制中心”窗口的对象视图中，选择 MRKJ 数据库文件夹，在该文件夹的右键快捷菜单中选择“除去”命令，如图 4.11 所示，在弹出的“确认”对话框中单击“确定”按钮，如图 4.12 所示。

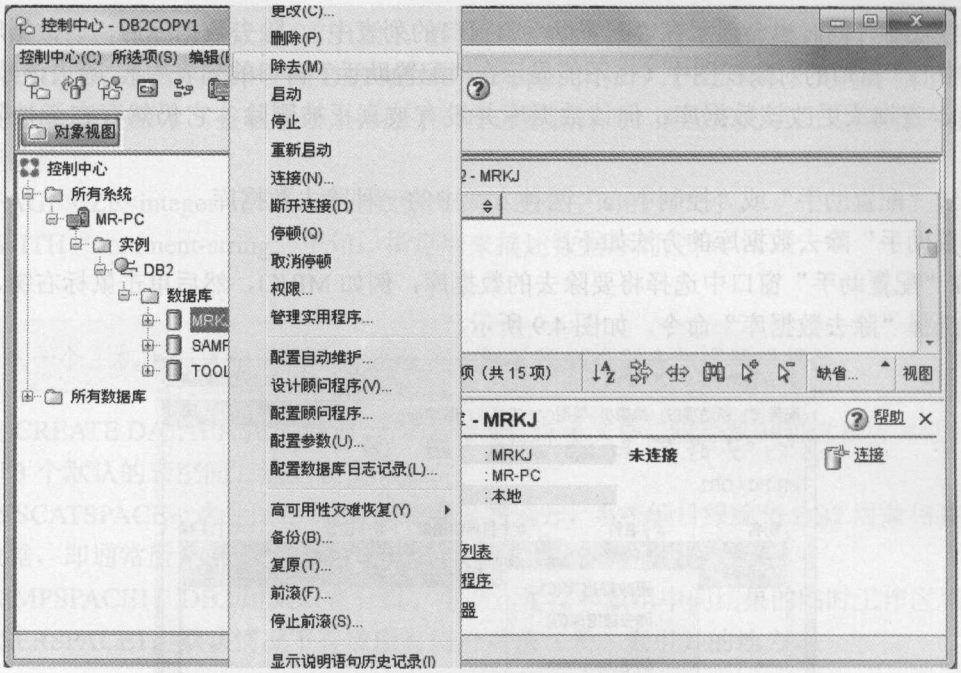


图 4.11　在“控制中心”除去 MRKJ 数据库

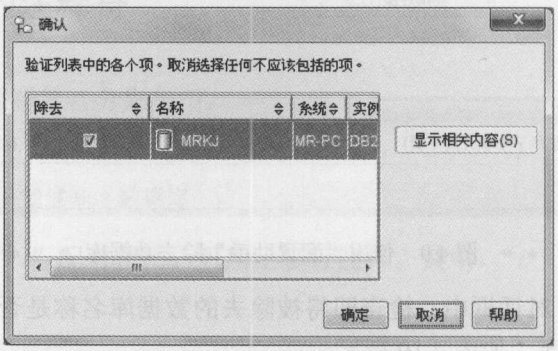


图 4.12　“确认”对话框

注意

如果已经除去某个数据库，则不能再次除去。对 DB2 初学者来说可以用以下方法练习编目数据库的操作，先用“配置助手”除去数据库，然后再用“配置助手”添加数据库；或者用“控制中心”再次除去数据库，然后再用“控制中心”添加数据库。

2．用界面方式添加本地数据库

这里先介绍如何添加本地已经被除去的数据库，本书将在第 14 章详细介绍如何添加网络数据库。

“添加”数据库：即添加本地已经存在但是被“除去”的数据库，或者添加的网络数据库已经存在于其他主机上，但是该数据库没有显示在本地“控制中心”窗口的对象视图中。添加数据库后，该数据库就会显示在本地“控制中心”窗口的对象视图中。在添加该数据库之前，不能从本地连接、查

询或更改该数据库。

用“配置助手”添加数据库的方法如下：

（1）在“配置助手”窗口的空白处单击鼠标右键，在弹出的快捷菜单中选择“使用向导来添加数据库”命令，如图 4.13 所示，从而启动“添加数据库向导”对话框。

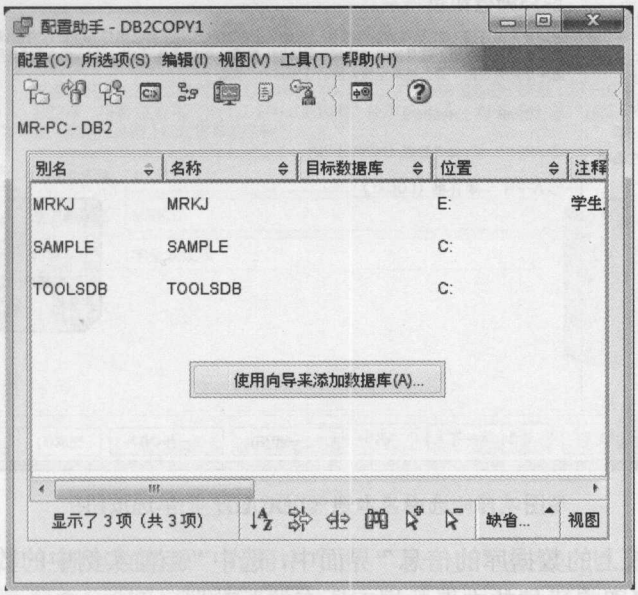


图 4.13　使用“添加数据库向导”对话框添加数据库

（2）在“添加数据库向导”对话框的“选择要如何设置连接”界面中，选中“手工配置与数据库的连接”单选按钮，单击“下一步”按钮，如图 4.14 所示。

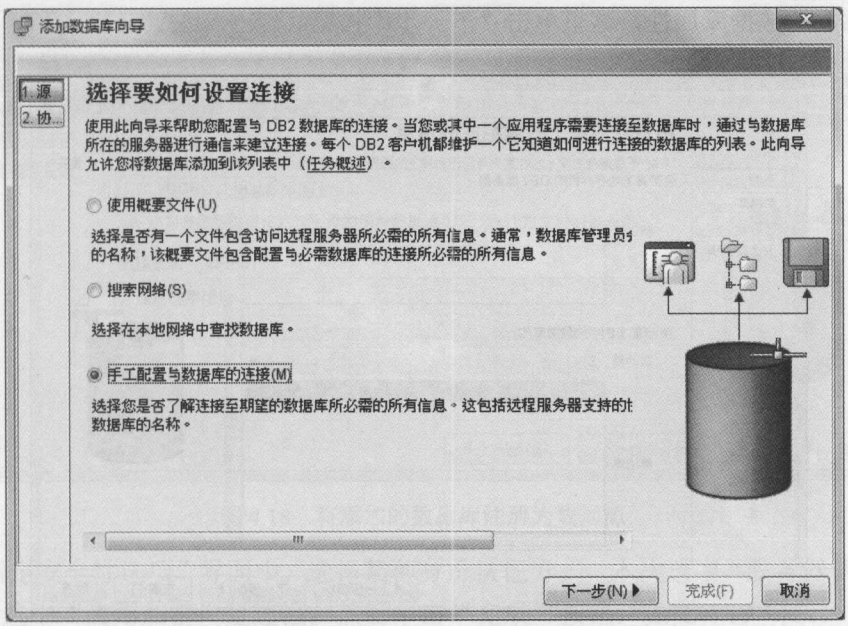


图 4.14　选中“手工配置与数据库的连接”单选按钮

（3）在“选择通信协议”界面中，选中“本地（LOCAL）”单选按钮，单击“下一步”按钮，如图 4.15 所示。

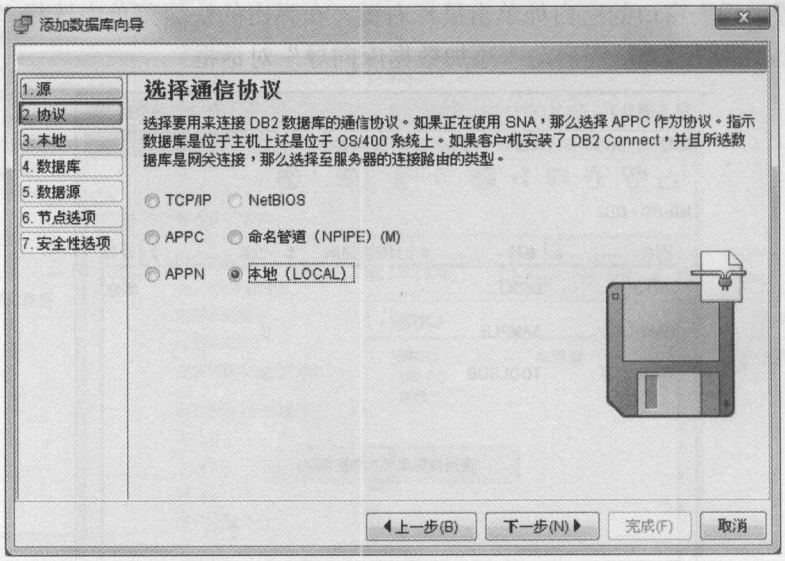


图 4.15　选中“本地（LOCAL）”单选按钮

（4）在“指定此系统上的数据库的信息”界面中，选中“当前实例中的数据库”单选按钮，在“驱动器”下拉列表框中选择即将添加的本地数据库所在的驱动器（例如：E:\）。选择完后，系统会自动在“数据库”列表中列出该驱动器上存储的所有数据库，然后选中准备添加的数据库（例如：MRKJ），如图 4.16 所示。如果要按照系统默认方式（别名、注释和认证方式等均为默认值）添加数据库，只需单击“完成”按钮即可完成添加数据库过程。如果想要设置数据库的属性，则单击“下一步”按钮，继续设置将要添加的数据库属性，这里单击“下一步”按钮。

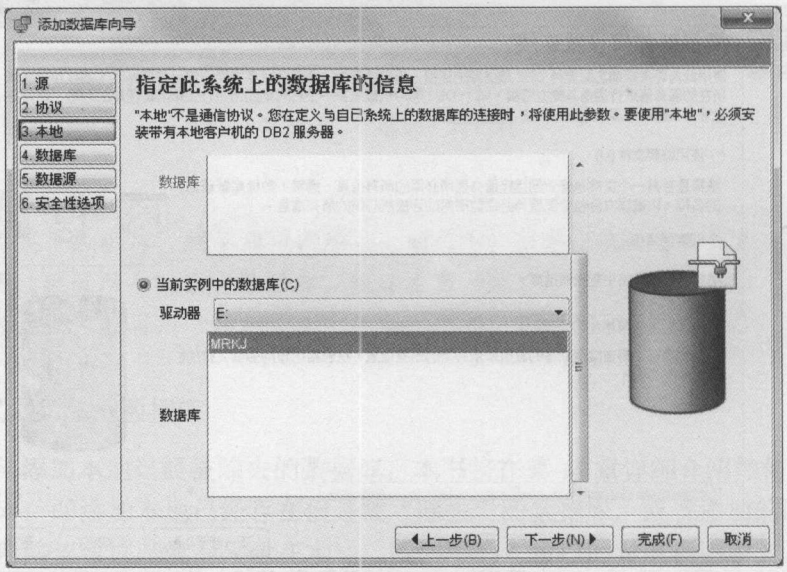


图 4.16　选择本地数据库

（5）在“指定要连接的数据库的名称”界面中的“数据库名称”和“数据库别名”文本框中会自动输入数据库的名称和别名，这里只需把数据库的别名由 MRKJ 改成 MRKJ1，在“注释”文本框中输入“学生成绩库”，然后单击“下一步”按钮，如图 4.17 所示。

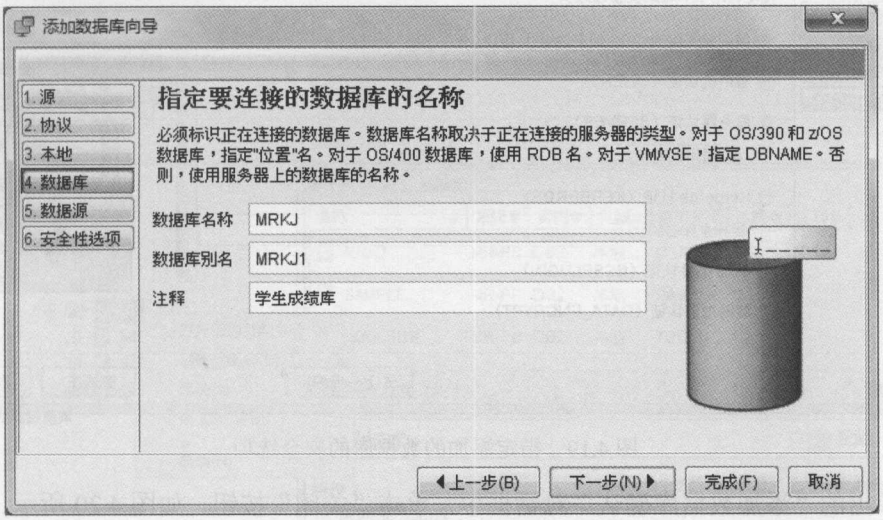


图 4.17　指定要连接的数据库名称、别名和注释

（6）在“将此数据库注册为数据源”界面中单击“下一步”按钮，以选择默认设置，如图 4.18 所示。系统会将 MRKJ 数据库注册为系统数据源 MRKJ1，添加该数据库之后，应用程序就可以通过 CLI/ODBC 驱动程序连接 MRKJ 数据库。

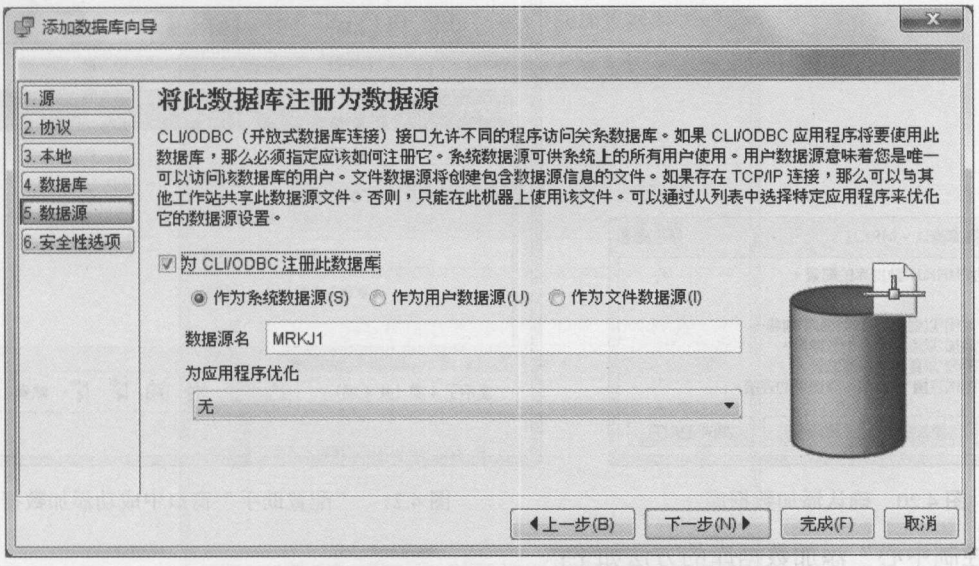


图 4.18　将添加的数据库注册为数据源

（7）在“指定安全性选项”界面中，选择某种身份认证方式，本次选中“服务器认证（SERVER）”单选按钮，并且选中“启用加密”复选框，然后单击“完成”按钮，如图 4.19 所示。

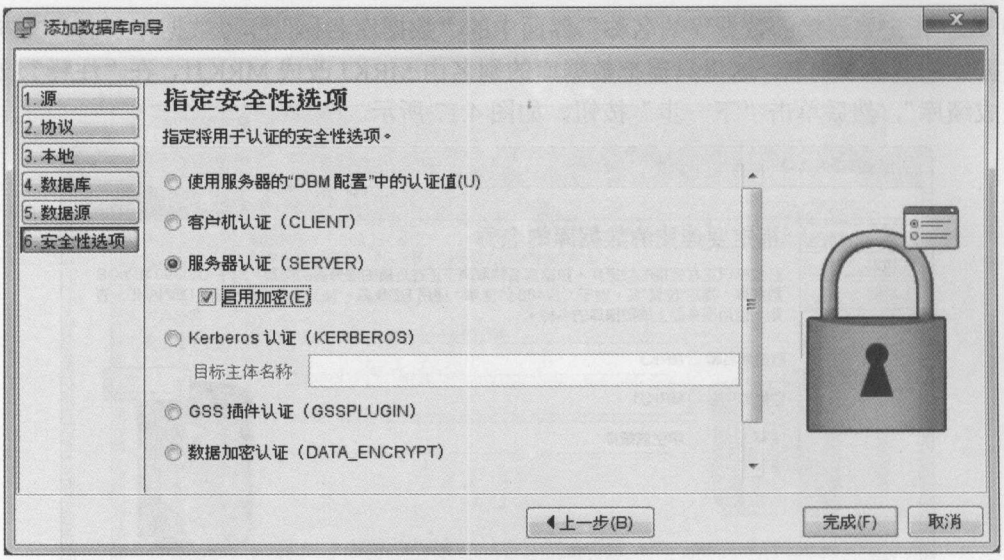


图 4.19　指定添加的数据库的安全选项

（8）在弹出的“添加数据库确认”对话框中，单击“关闭”按钮，如图 4.20 所示。这时就已经成功添加数据库，回到“配置助手”窗口即可看到添加的数据库 MRKJ1，如图 4.21 所示。

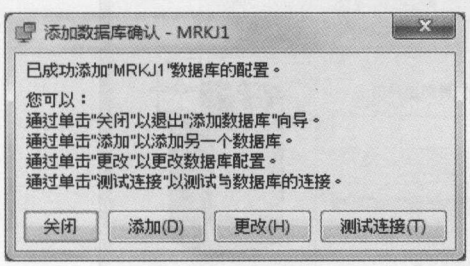


图 4.20　确认添加数据库

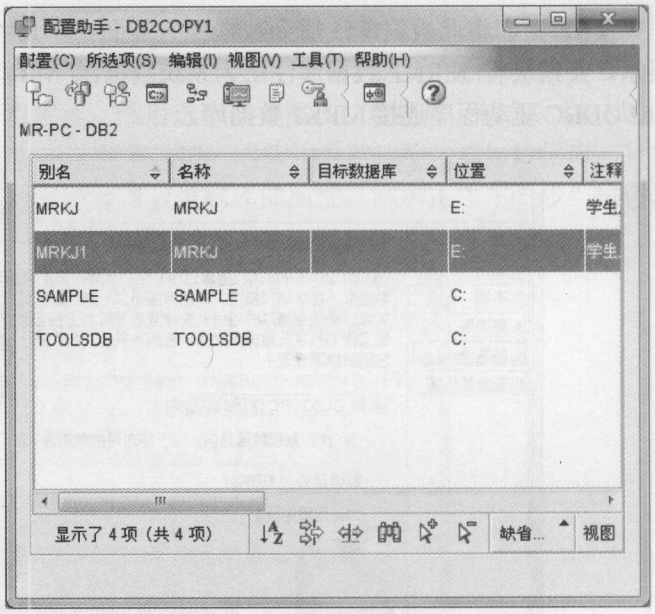


图 4.21　“配置助手”窗口中成功添加数据库

用“控制中心”添加数据库的方法如下：

（1）在“控制中心”窗口对象视图中选择“所有系统”→“服务器名称”（这里是 MR-PC）→“实例”→“实例名”（这里是 DB2）→“数据库”文件夹，在该文件夹上单击鼠标右键，在弹出的快捷菜单中选择“添加”命令，如图 4.22 所示。

（2）在弹出的“添加数据库”对话框中，在“驱动器”下拉列表框中选择“E:”，在“数据库名

称”下拉列表框中选择 MRKJ，在“别名”文本框中输入 MRKJ1，在“注释”文本框中输入“学生成绩库”，在“认证”选项区域的“类型”下拉列表框中选择“SERVER ENCRYPT”，然后单击“确定”按钮完成数据库的添加，如图 4.23 所示。

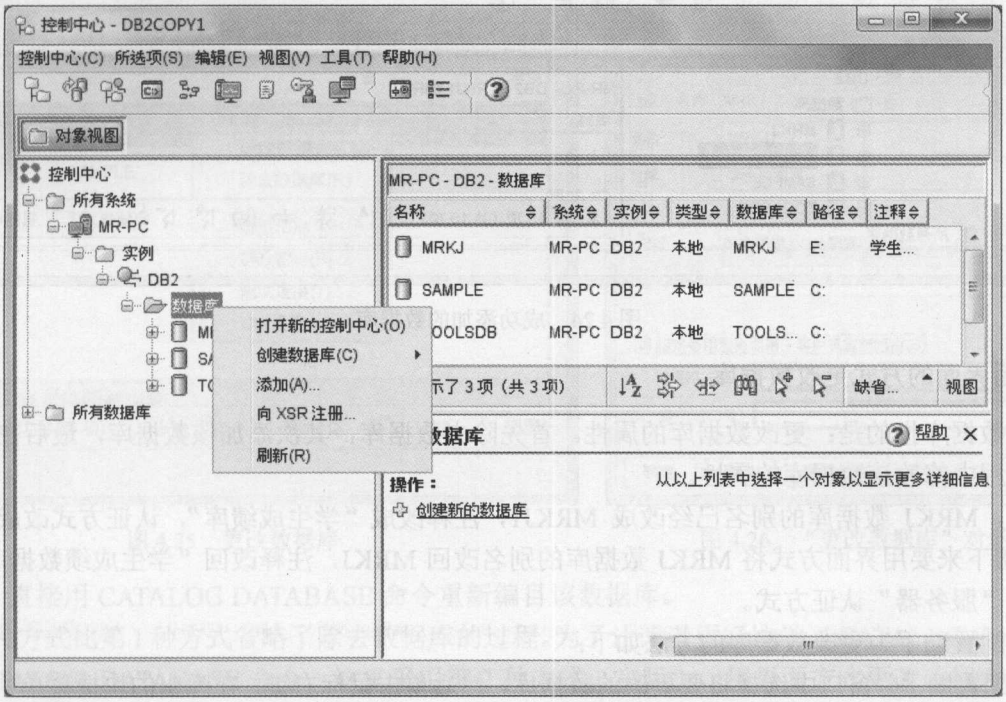


图 4.22　使用“控制中心”添加数据库

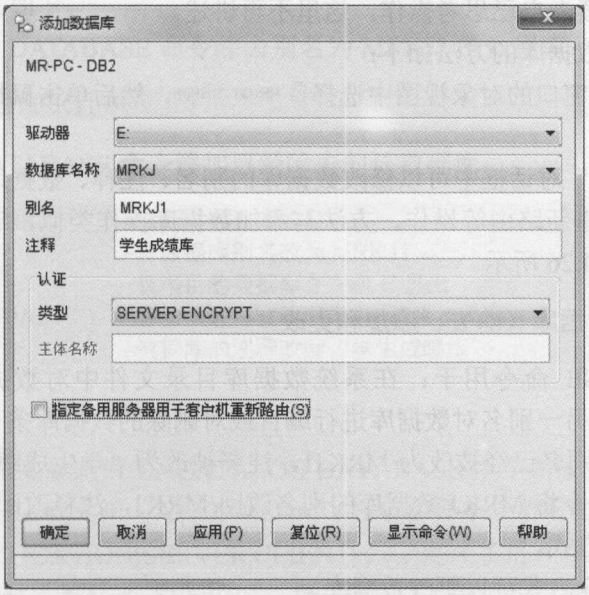


图 4.23　“添加数据库”对话框

（3）数据库添加成功后，会在“控制中心”窗口的对象视图中看到添加的数据库 MRKJ1，如图 4.24

所示。

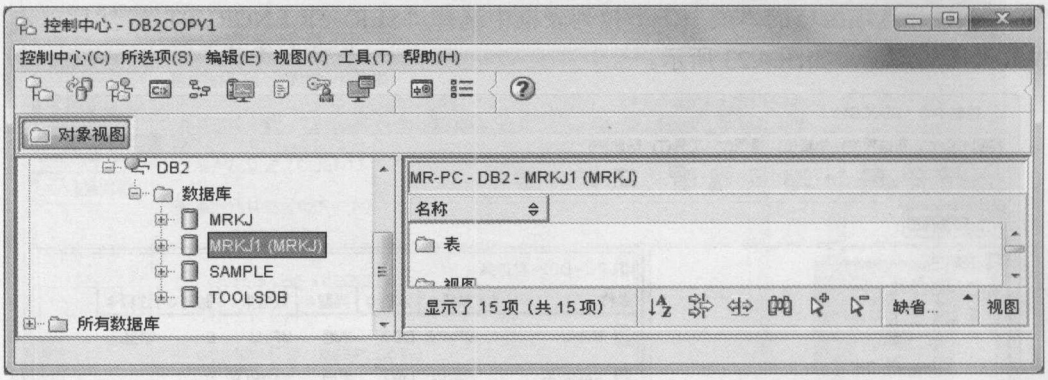


图 4.24　成功添加的数据库

3．用界面的方式更改数据库

更改数据库指的是：更改数据库的属性。首先除去数据库，其次添加该数据库，最后在添加该数据库的过程中修改该数据库的属性。

例如，MRKJ 数据库的别名已经改成 MRKJ1，注释改成“学生成绩库”，认证方式改成“服务器加密”。接下来要用界面方式将 MRKJ 数据库的别名改回 MRKJ，注释改回“学生成绩数据库”，认证方式改回“服务器”认证方式。

用“配置助手”更改数据库的方法如下：

在“配置助手”窗口中选择将要更改的数据库，然后单击鼠标右键，在弹出的快捷菜单中选择“更改数据库”命令，如图 4.25 所示。然后打开“更改数据库的向导”对话框，其余的操作过程与添加数据库的操作过程类似，请读者自己思考操作，这里不再讲述。

用“控制中心”更改数据库的方法如下：

（1）在“控制中心”窗口的对象视图中选择 MRKJ1 (MRKJ)，然后单击鼠标右键，在弹出的快捷菜单中选择“更改”命令。

（2）在“更改数据库”对话框中可以修改数据库的别名、注释、数据库认证方式以及控制是否指定备用服务器用于客户机重新路由等操作，方法与添加数据库操作类似，这里不再讲述。确认无误后单击“确定”按钮，如图 4.26 所示。

4．用命令方式编目数据库（除去、添加和更改）

CATALOG DATABASE 命令用于：在系统数据库目录文件中对数据库进行编目。CATALOG DATABASE 命令允许使用另一别名对数据库进行编目或对删除的数据库条目进行编目。

假如 MRKJ 数据库的别名已经被改为 MRKJ1，注释被改为“学生成绩库”，认证方式已经被改为“服务器加密”。现要用命令将 MRKJ 数据库的别名改回 MRKJ，注释改回“学生成绩数据库”，认证方式改回“服务器”认证方式。

分析：有以下两种命令方式可以更改数据库。

（1）首先用 UNCATALOG DATABASE 命令除去数据库，然后用 CATALOG DATABASE 命令添加已经除去的该数据库。

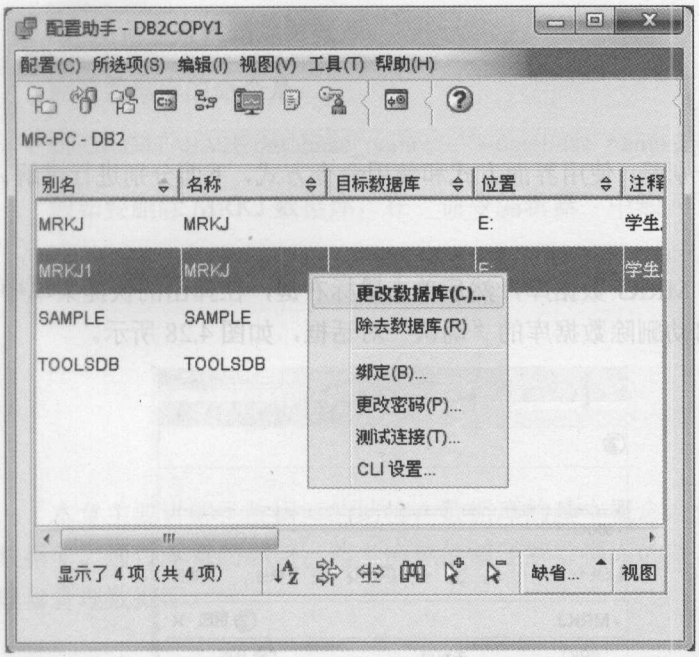


图 4.25　更改数据库

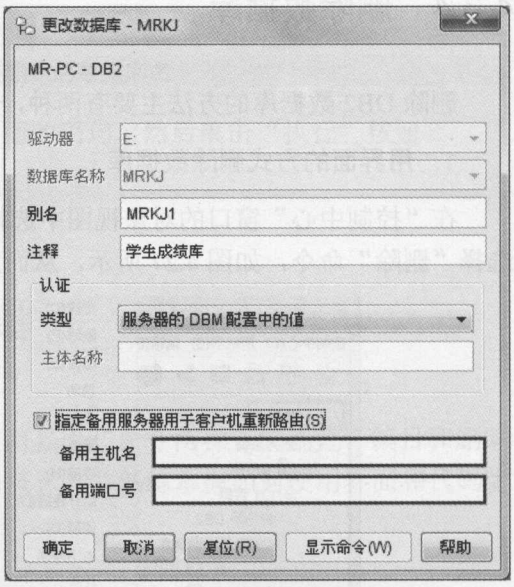


图 4.26　“更改数据库”对话框

（2）直接用 CATALOG DATABASE 命令重新编目该数据库。

第 2 种方式比第 1 种方式省略了除去数据库的过程。为了让读者更好地学习 CATALOG DATABASE 和 UNCATALOG DATABASE 命令，这里采用第 2 种方式。实际上，用界面方式更改数据库时，系统也是用第 2 种方式更改数据库。

编目数据库的操作方法如下。

（1）用 UNCATALOG DATABASE 命令除去别名为 MRKJ1 的数据库：

```
UNCATALOG DATABASE MRKJ1;
```

（2）用 CATALOG DATABASE 命令添加已经除去的该数据库：

```
CATALOG DATABASE MRKJ          --重新对数据库编目
AS  MRKJ1                      --将数据库别名改为 MRKJ1
ON E                           --被编目的数据库在主机 E 盘上
AUTHENTICATION  SERVER         --身份认证方式改回 SERVER
WITH'学生成绩库';              --数据库的注释改回 '学生成绩库'
```

说明

（1）除去数据库的本质就是用 UNCATALOG DATABASE 命令删除数据库编目。

（2）添加数据库本质上就是用 CATALOG DATABASE 命令重新对已除去的数据库编目。

（3）子句 AUTHENTICATION[SERVER|CLIENT|……]定义了用户在连接数据库时，在什么地方以及怎样进行数据库认证。默认为 SERVER，指在包含了目标数据库的服务器上认证。CLIENT 表示认证在调用应用的节点或者分区上进行。

4.3.4 删除数据库

删除 DB2 数据库的方法主要有两种，分别是：使用界面方式和使用命令方式，下面分别进行讲解。

1. 用界面的方式删除数据库

在“控制中心”窗口的对象视图中选择 MRKJ 数据库，然后单击鼠标右键，在弹出的快捷菜单中选择“删除”命令，如图 4.27 所示，从而启动删除数据库的“确认”对话框，如图 4.28 所示。



图 4.27 删除数据库

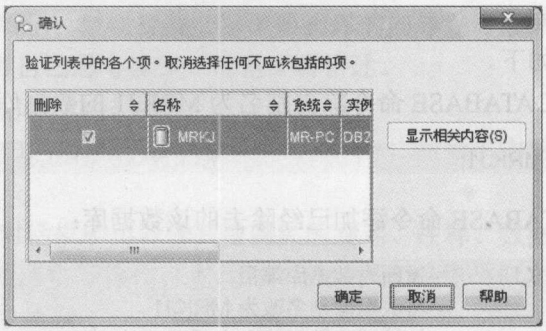


图 4.28 “确认”对话框

注意

“删除”命令是彻底删除数据库数据，无法恢复。而“除去”命令表示仅将数据库从系统目录中删除，如果需要还可以恢复。这里建议最好使用“控制中心”对象视图中“所有系统”→“服务器名称”→“实例”→“实例名”→“数据库”→MRKJ 删除 MRKJ 数据库。而不要在“控制中心”对象视图中的“所有数据库”→MRKJ 文件夹上右击，然后在弹出的快捷菜单中选择“删除”命令来删除 MRKJ 数据库。如果这样，删除 MRKJ 数据库后就不能删除 MRKJ 数据库对应的数据库工具目录信息。如果想重新创建 MRKJ 数据库或利用备份文件在原数据库所在的目录下恢复 MRKJ 数据库就会报错且无法创建或恢复 MRKJ 数据库。

2．用命令方式删除数据库

删除数据库的命令是：

```
DROP DATABASE database_name;      --database_name 是要删除的数据库名
```

假如要删除 MRKJ 数据库，在“命令编辑器”中输入下面的语句，然后单击“执行”按钮▶：

```
DROP DATABASE MRKJ;               --删除 MRKJ 数据库
```

4.4　小　　结

本章主要讲解了数据库的结构，数据库的基本概念，使用界面和命令两种方式创建、编目和删除数据库。通过本章的学习，读者应该能够了解数据库的基本概念，并熟练掌握如何使用界面和代码管理器管理数据库。

4.5　实践与练习

（1）DB2 数据库的命名需要注意什么？（**答案位置：光盘\TM\sl\4\1**）

（2）如何在 DB2 语句中添加注释文字？（**答案位置：光盘\TM\sl\4\2**）

第5章

管理表空间和表

（视频讲解：53分钟）

本章主要讲解如何使用“控制中心”和“命令编辑器”对表空间和表进行创建、修改及删除等操作，通过本章的学习，读者能够掌握创建、管理表空间和表的方法。

通过阅读本章，您可以：

- 掌握表空间的创建、修改和删除
- 掌握表的创建、修改和删除

5.1　表空间的创建、修改和删除

创建表之前要给表准备一个容器来存储表，这个容器就是表空间。

DB2 的表空间按管理方式分为两种：系统管理空间（System Management Space，SMS）和数据库管理空间（Database Management Space，DMS）。

DB2 的表空间按类型分为：规则表空间、大对象表空间、系统临时表空间、用户临时表空间。

- ☑ 规则表空间中包含用户数据的表。默认用户表空间名为 USERSPACE1，索引也存储在规则表空间中，另外系统目录表也放在规则表空间中。默认的系统目录表空间名为 SYSCATSPACE。
- ☑ 大对象表空间。
- ☑ 系统临时表空间用来存储各种数据操作（排序、重组表、创建索引、连接表）中所需的内部临时数据，虽然可以创建任意多个系统临时表空间，但建议用户只根据大多数表所使用的页大小创建一个，默认系统临时表空间名为 TEMPSPACE1。
- ☑ 用户临时表空间用来存储已说明全局临时表（已说明全局临时表存储的是应用程序临时数据）。用户临时表空间不是在数据库创建时默认创建的。

SMS 每个容器是操作系统的文件空间中的一个目录，DMS 每个容器是一个固定的、预分配的文件，或是物理设备。SMS 的管理比较简单，由操作系统自动管理，空间的大小随数据量的变化由系统自动调整。DMS 是由数据库管理的，空间大小在创建时确定，空间不够时要手工添加或删除部分数据以释放空间。

DMS 与 SMS 的比较如表 5.1 所示。

表 5.1　DMS 与 SMS 之间的比较

特　性	SMS	DMS
是否条带化（Strping）	是	是
默认类型	Version 8	无
对象管理	操作系统	DB2
空间分配	按需增长/收缩	预先分配；大小可以收缩和增长，但是需要 DBA 干预
管理的简便性	最好；很少需要调优，甚至不需要调优	好，但是需要一些调优
性能	不太好	很好；可通过裸设备多获得 5%到 10%的利益

SMS 表空间可以简化管理，DMS 表空间可以提高性能，除此之外，这两种存储模式之间最显著的差异是表空间的最大容量值。在使用 SMS 表空间时，DBA 最多只能在表空间中存放 64GB（4KB 页大小）的数据。将页面大小改为 32KB，可以将这个限制扩大到 512GB，但代价是每个页面上的可用空间可能会更少。在 DB2 V9.7 中，DMS 模型会将大型表空间的限制扩大到 8TB（4KB 页面大小的情况下）。如果将页面大小改为 32KB，可用空间可以增长到 64TB。尽管还有让表大小突破 64GB 限制的其他方法，但是最简单的方法可能是一开始就使用 DMS 大型表空间。

5.1.1 创建表空间

下面用两种方法创建表空间，分别是使用界面方式创建表空间和使用命令方式创建表空间。

1. 用界面方式创建表空间

创建表空间的步骤如下：

（1）在“控制中心”窗口的对象视图中选择 MRKJ 数据库下的“表空间”文件夹，在该文件夹上单击鼠标右键，在弹出的快捷菜单中选择“创建”命令，从而启动“创建表空间向导”对话框，如图 5.1 所示。

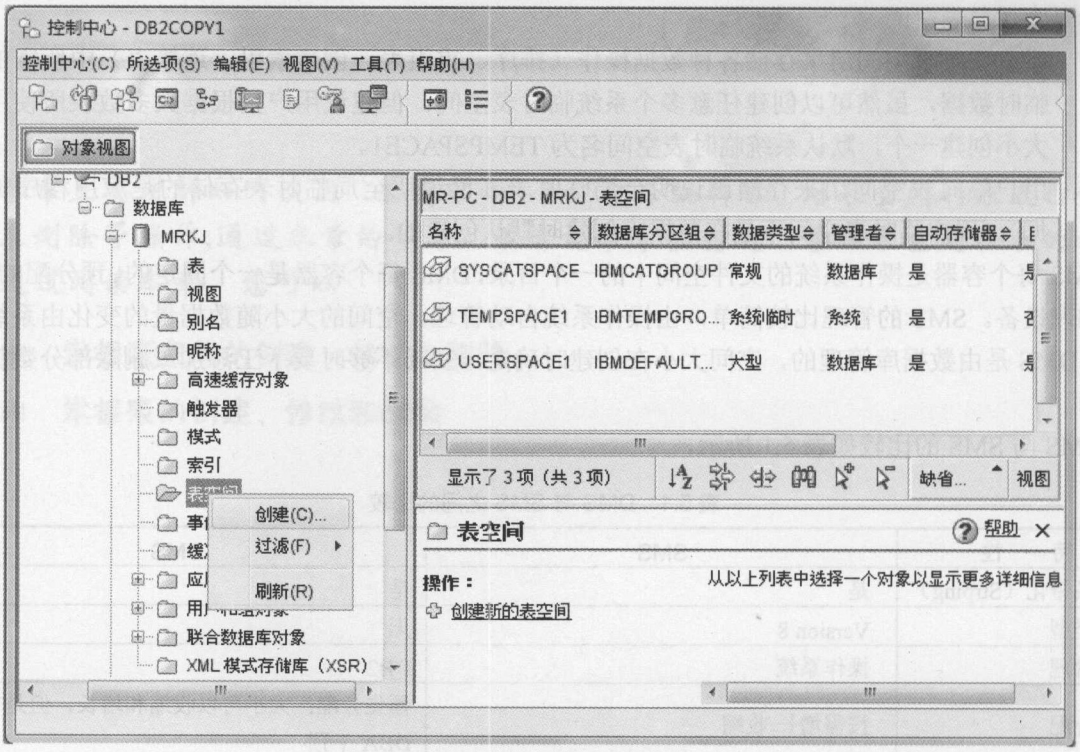


图 5.1 创建表空间菜单

（2）在“创建表空间向导”对话框中的“指定新表空间的名称”界面中，在“表空间名”文本框中输入“CUSTOMDATASPACE1”，在“注释”文本框中输入“自定义数据空间 1”，选中“让 DB2 管理存储器（自动存储器）”单选按钮（如果选中“我想手工管理存储器”单选按钮，则会增加两个步骤，分别是“空间管理”和“容器”），如图 5.2 所示，最后单击“下一步”按钮。

（3）在“指定想要创建的表空间的类型”界面中选择默认选项“常规”，可用于存储除临时表之外的索引和任何类型的数据。“大型”单选按钮，可用于存储大型表列或 LOB 表列中的数据（长型文本、音频、视频或图像），也可以存储索引。“系统临时”单选按钮在创建数据库时已经自动创建了表

空间，为排序、连接和其他操作提供数据库管理器空间。“用户临时”单选按钮为已声明的临时表提供用户应用程序空间，如图 5.3 所示。创建表空间向导的其他步骤一般都不需要设置。直接单击“完成”按钮结束创建，系统创建完表空间后会弹出命令成功完成的消息。

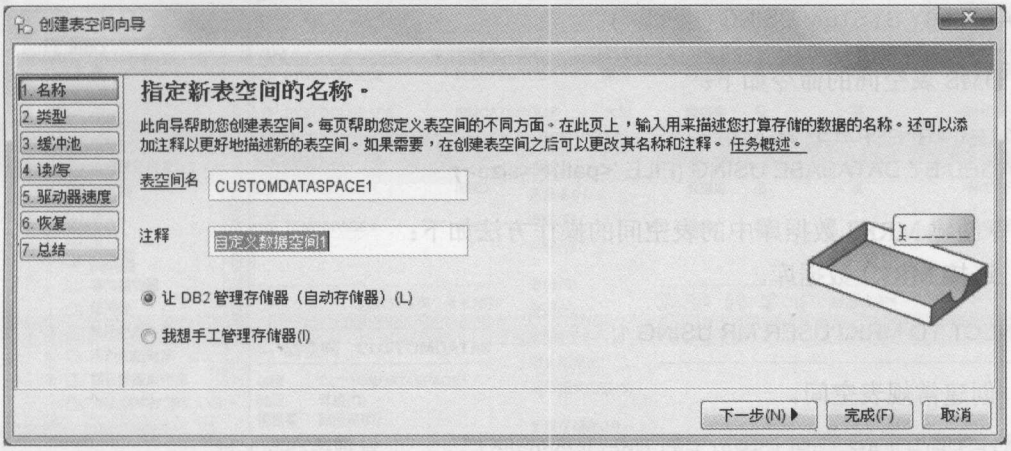


图 5.2　“指定表空间名称”界面

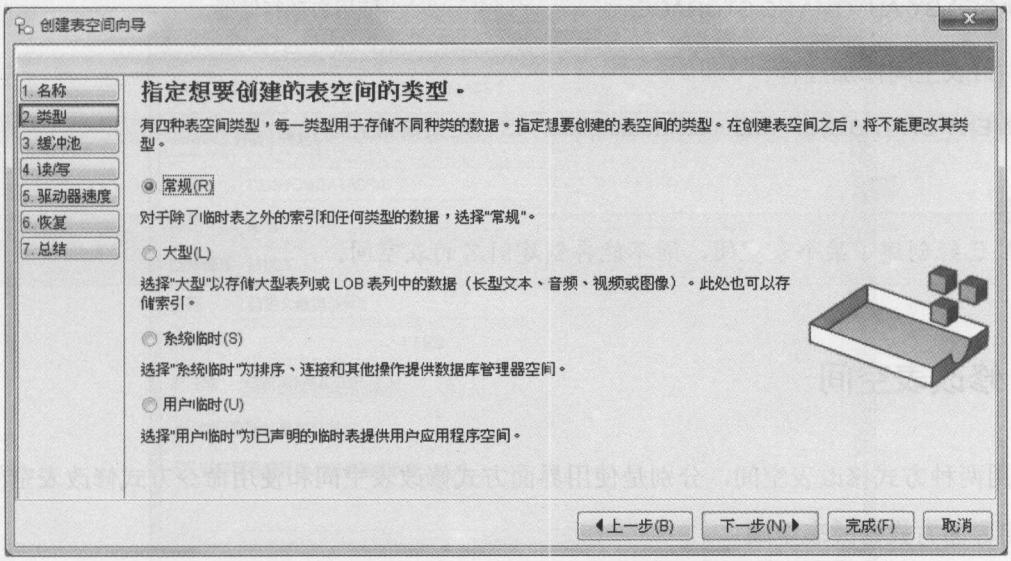


图 5.3　“指定想要创建的表空间类型”界面

2. 用命令方式创建表空间

当创建数据库时，会自动创建 3 个初始表空间。这 3 个初始表空间的页的大小是使用 CREATE DATABASE 命令时建立或接受的默认值。此默认值还表示将来所有 CREATE BUFFERPOOL 和 CREATE TABLESPACE 语句的默认页大小。如果在创建数据库时不指定页大小，那么默认页大小是 4KB。如果在创建表空间时不指定页大小，那么默认页大小是创建数据库时设置的页大小。

对于非自动存储器表空间，在创建表空间时，必须要知道将引用的容器的设备名或文件名。另外，也必须知道与要分配给表空间的每个设备名或文件名相关联的空间量。对于自动存储器表空间，数据

库管理器将根据与数据库关联的存储路径将容器指定给表空间。

创建 SMS 表空间的命令如下：

```
CREATE TABLESPACE <NAME>
MANAGED BY SYSTEM USING ('<path>')
```

创建 DMS 表空间的命令如下：

```
CREATE TABLESPACE <NAME>
MANAGED BY DATABASE USING (FILE '<path>'<size>)
```

用命令创建 MRKJ 数据库中的表空间的操作方法如下：

（1）连接 MRKJ 数据库。

```
CONNECT TO MRKJ USER MR USING 1;
```

（2）创建常规表空间。

```
CREATE REGULAR TABLESPACE customdataspace1          --常规表空间名称
PAGESIZE 4K                                         --页大小
MANAGED BY AUTOMATIC STORAGE;                       --使用自动存储器
```

（3）给表空间添加注释。

```
COMMENT ON TABLESPACE customdataspace1 IS'自定义数据空间 1';
```

注意

如果已经创建了某个表空间，则不能再创建同名的表空间。

5.1.2 修改表空间

下面用两种方式修改表空间，分别是使用界面方式修改表空间和使用命令方式修改表空间。

1. 用界面方式修改表空间

修改表空间的步骤如下：

（1）在“控制中心”窗口对象视图中选择 MRKJ 数据库下的“表空间”文件夹，在内容窗格会显示 MRKJ 数据库中的全部表空间，选择要修改的表空间“CUSTOMDATASPACE1”，然后单击鼠标右键，在弹出的快捷菜单中选择“改变”命令，如图 5.4 所示。

（2）在“改变表空间”对话框中，包含“表空间”、“容器”和“性能”3 个选项卡。可以改变注释、切换缓冲池、增减容器和变更表空间预取大小，如图 5.5 所示，其中灰色部分为不可修改项。修改之后，单击“确定”按钮则完成修改。

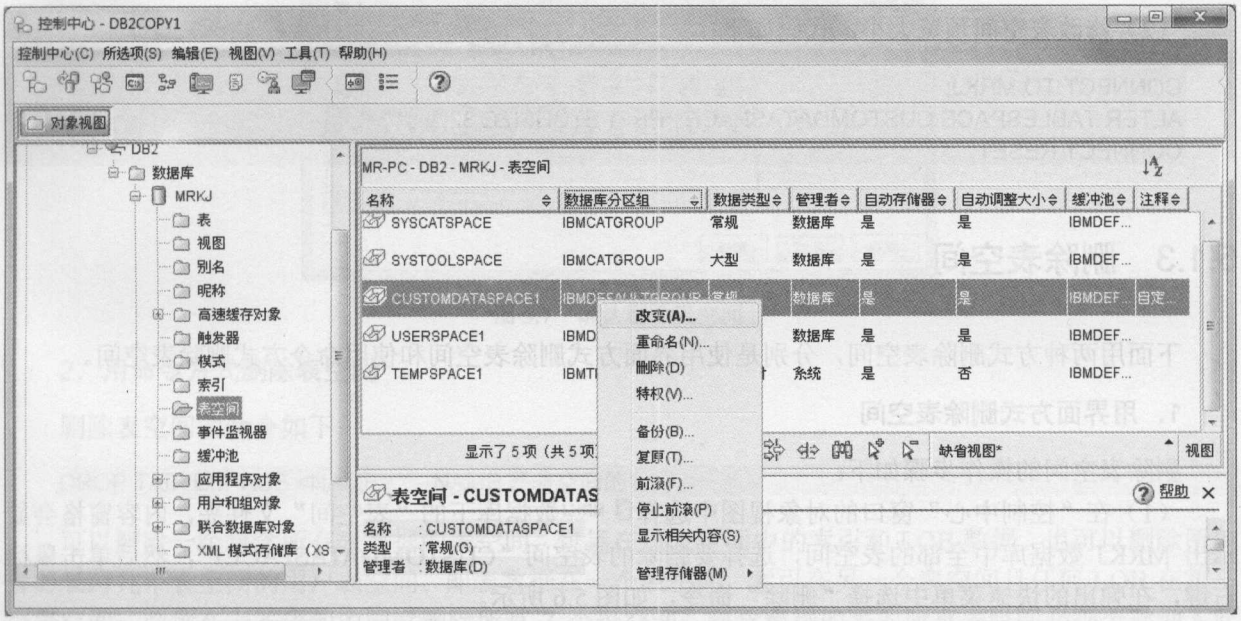


图 5.4　修改表空间的命令

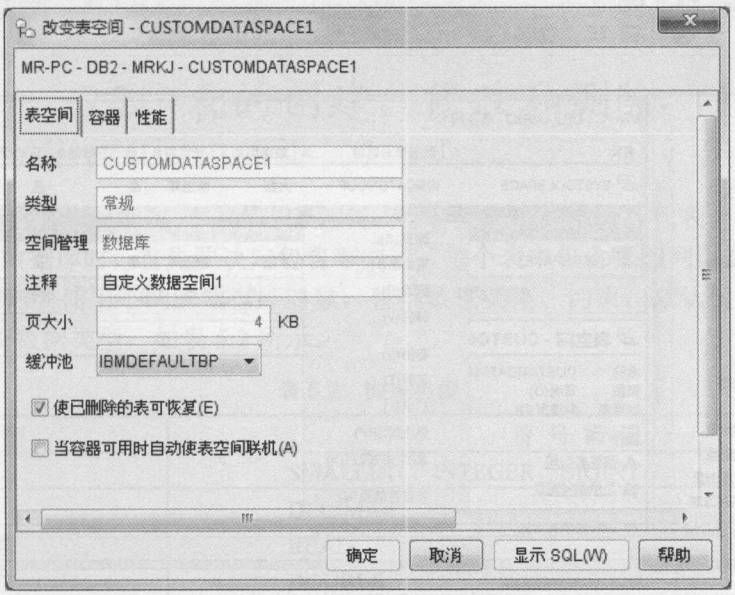


图 5.5　修改表空间参数

2. 用命令方式修改表空间

在“命令编辑器”中输入如下命令，然后单击“执行”按钮 ▶ 即可修改表空间。

（1）修改表空间的注释，代码如下：

```
CONNECT TO MRKJ;
COMMENT ON TABLESPACE CUSTOMDATASPACE1 IS '自定义数据空间 1';
CONNECT RESET;
```

（2）修改表空间预览大小，代码如下：

```
CONNECT TO MRKJ;
ALTER TABLESPACE CUSTOMDATASPACE1 PREFETCHSIZE 32;
CONNECT RESET;
```

5.1.3 删除表空间

下面用两种方式删除表空间，分别是使用界面方式删除表空间和使用命令方式删除表空间。

1. 用界面方式删除表空间

删除表空间的操作步骤如下：

（1）在“控制中心”窗口的对象视图中选择 MRKJ 数据库下的“表空间”文件夹，内容窗格会显示出 MRKJ 数据库中全部的表空间，选择要删除的表空间“CUSTOMDATASPACE1”，然后单击鼠标右键，在弹出的快捷菜单中选择“删除”命令，如图 5.6 所示。

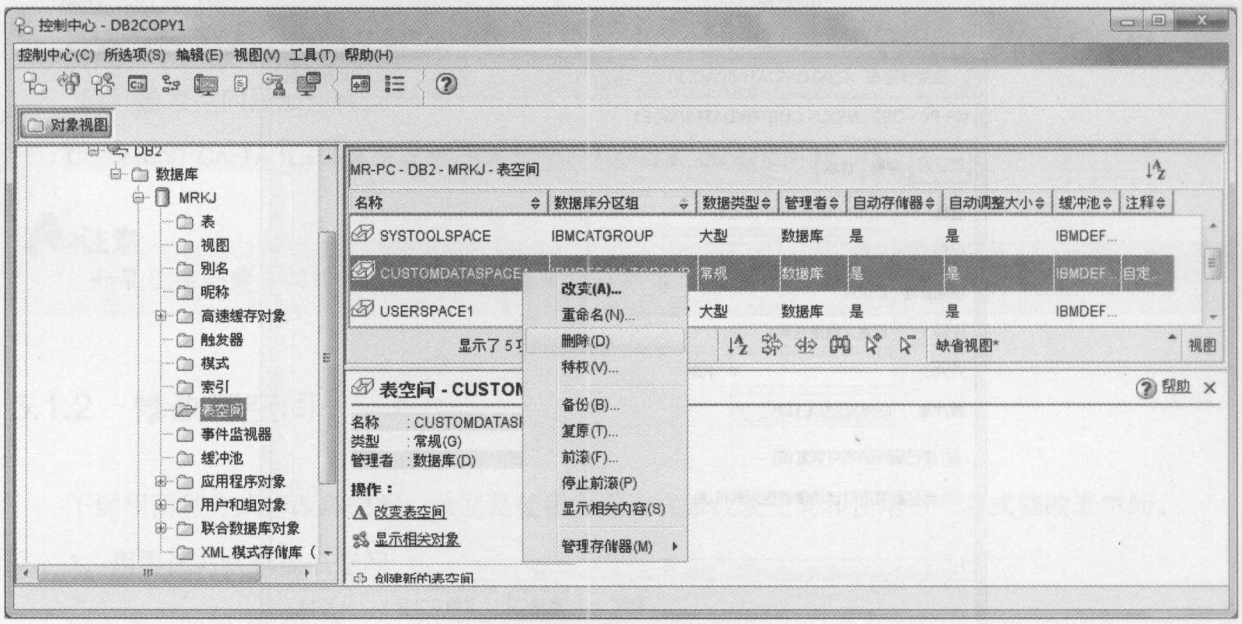


图 5.6 删除表空间命令

（2）在弹出的“确认”对话框中需要验证列表的对象是否是需要删除的表空间。验证无误后单击“确定”按钮完成删除，如图 5.7 所示。

注意

单击“显示相关内容”，来显示该表空间包含的表或索引。一旦删除表空间，表空间包含的数据也将被删除。

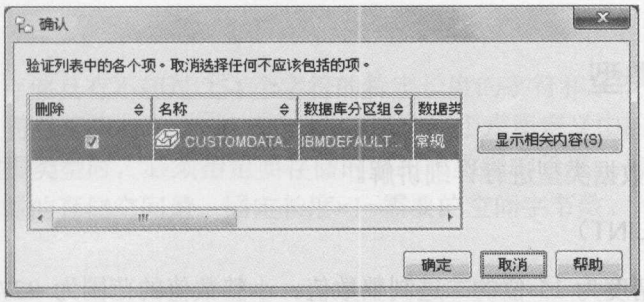


图 5.7　确认删除表空间

2. 用命令方式删除表空间

删除表空间的命令如下：

```
DROP TABLESPACE <NAME>   --NAME 是表空间的名称
```

可以删除一个包含所有表数据的表空间，包括在该表空间中的索引和 LOB 数据。也可以删除所包含的表跨几个表空间的用户表空间。即表数据在一个表空间，索引在另一个表空间且任何 LOB 在第三个表空间。必须在一条语句中同时删除所有 3 个表空间。包含跨越的表的所有表空间必须全部纳入此单条语句中，否则该删除请求将失败。

5.2　表的创建、修改和删除

表是存放数据库数据的对象，表中的数据组织成行、列的形式。每行代表一条数据，每列代表数据的一个属性。设计数据库时，首先要决定包含哪些表，每个表中包含哪些列以及每列的数据类型等。当创建和使用表时，需要使用不同的数据库对象，包括数据类型、约束、默认值、触发器和索引等。下面介绍 DB2 的系统数据类型，如表 5.2 所示。

表 5.2　数据类型

数 据 类 型	符 号 标 识
整数型	SMALLINT、INTEGER、BIGINT
精确数值型	DECIMAL
单精度浮点型	REAL
双精度浮点型	DOUBLE
货币型	DECFLOAT
固定长度字符型	CHARACTER
可变长度字符串	VARCHAR、LONG、VARCHAR
字符大对象型	CLOB
二进制大对象型	BLOB
日期时间类型	DATE、TIME、TIMESTAMP
图形串数据	GRAPHIC、VARGRAPHIC、LONG VARGRAPHIC
XML 类型	XML

5.2.1 DB2 数据类型

下面对 DB2 的系统数据类型进行详细讲解。

1. 小整数（SMALLINT）

小整数型用于存储精度为 15 位的二进制整数值。小整数值的范围为-32768~+32767。小整数数据类型使用可能的最小存储空间量来存储数值（存储每个值需要 2 个字节的空间）。符号标识 SMALLINT 用于声明小整数列。

2. 整数（INTEGER）

整数型用于存储精度为 31 位的二进制整数值。虽然整数数据类型需要的存储空间是小整数数据类型的 2 倍（存储每个值需要 4 个字节的空间），但它的取值范围要大很多。整数值的取值范围是-2147483648~+2147483647。符号标识 INTEGER 和 INT 用于声明整数列。

3. 大整数（BIGINT）

大整数型用于在支持 64 位整数的平台上存储精度为 63 位的二进制整数值。处理作为大整数存储的较大数字比处理作为十进制值存储的类似数字效率要高很多。使用大整数值执行的计算比使用实值型或双精度值执行的计算更精确。

大整数型需要的存储空间是小整数数据类型的 4 倍（存储每个值需要 8 个字节的空间）。大整数的取值范围为-9223372036854775808~+9223372036854775807。符号标识 BIGINT 用于声明大整数列。

4. 十进制

十进制类型用于存储同时包含整数部分和小数部分的数字，各部分组合在一起并采用压缩十进制格式存储。每次声明十进制数据类型时，必须指定精度和小数位。十进制的精度范围为 1~31。可以使用下面的等式计算存储十进制值所需的存储空间量：精度/2+1=需要的空间字节数。

例如，DECIMAL（5,2）值需要 3 个字节的存储空间（5/2=2.5（截断为 2）；2+1=3）。

5. 单精度浮点

单精度浮点类型用于 32 位实数近似值。虽然单精度浮点数据类型和整数数据类型需要的存储空间量相同（存储每个值需要 4 个字节的空间），但单精度浮点数的范围要大很多：10E-38~10E+38。符号标识：REAL 和 FLOAT 可用于在表定义中声明单精度浮点列，但是采用 FLOAT 时指定列的长度必须在 1~24 之间。FLOAT 可用来同时表示单精度和双精度浮点数据类型；指定的长度确定使用的实际数据类型。

6. 双精度浮点

此数据类型用于存储 64 位实数近似值。虽然双精度浮点数据类型需要的存储空间量与大整数数据类型相同（存储每个值需要 8 个字节的空间），但双精度浮点数的范围是可能的最大值：-1.79760E+308 到-2.225E-307、0 和 2.225E-307 到-1.79769E+308。

7．固定长度字符串

固定长度字符串用于存储具有不超过 254 个字符的特定长度的字符和字符串。如果没有指定长度，那么就认为是 1 个字节。符号标识 CHARACTER 和 CHAR 用于声明定义中的固定长度字符串列；每次声明固定长度字符串数据类型时，必须指定要存储的字符串数据的长度。可以使用下面的等式确定存储固定长度字符串值所需的存储空间量：固定长度×1=需要的空间字节数。

注意

使用固定长度字符串数据类型时，如果数据的实际长度比定义列时指定的长度要小很多，那么可能会浪费存储空间。

8．可变长度字符数据

此数据类型用于存储长度变化时的字符串数据。变长字符串数据的长度最大可为 32672 个字符；但是，对于使用 4KB 页的表空间中的表，变长字符串数据的长度不能超过 4092 个字符；对于使用 8KB 页的表空间中的表，变长字符串数据的长度不能超过 8188 个字符，依此类推，页大小最多为 32KB。因为默认状态下表空间是用 4KB 页创建的，如果要使用变长字符串数据类型来存储包含多于 4092 个字符的字符串，那么必须使用更大的页显式创建表空间。

注意

表行中还必须要有足够的空间来容纳字符串数据。也就是说，字符串数据的存储器需求必须包括表中其他列的存储器需求，并且需要的存储空间总量不能超过表空间的页大小。更新变长字符串数据值后，如果新值大于原始值，那么包含该值的记录将移到表中的另一页，这种记录称为指针记录。

符号标识 CHARACTER VARYING、CHAR VARYING 和 VARCHA 可用于在表定义中声明变长字符串系列。在定义变长字符串列时，必须在声明中指定应存储在该列中的最大字符数。以后存储在该列中的字符串数据值可以小于或等于指定的最大长度；如果它们更长，那么将不会存储它们，并且会返回错误。

使用下面的等式确定存储变长字符串值所需的存储空间量：（字符串长度×1）+4=需要的空间字节数。即如果使用 VARCHAR（30）数据类型存储了包含 30 个字符的字符串，那么该特定值将需要 34 个字节的存储空间（使用此数据类型的所有字符串的长度必须小于或等于 30 个字符串）。

9．可变长度长字符串数据

变长长字符串数据类型也用于存储长度变化的字符串数据。此数据类型用于将长度小于或等于 32700 个字符的字符串数据存储在使用 4KB 页的表空间的表中。

符号标识 LONG VARCHAR 用于在表定义中声明变长长字符串列。可以使用以下等式确定存储变长长字符串值所需的存储空间量：（字符串长度×1）+24=需要的空间字节数。

注意

在表定义中声明列时，可以将 FOR BIT DATA 子句与任何字符串数据类型配合使用。如果使用此子句，那么在数据交互操作期间，不会执行代码页转换，并且会将数据本身视为二进制（位）数据进行比较。

10. 字符大对象（CLOB）

CLOB（字符大对象）值的长度最多为 2GB 字节（2147483647 字节），如果只指定了 n，那么 n 的值就是最大长度。如果指定了 nK，那么最大长度就是 n×1024（n 的最大值为 2097152）。如果指定了 nM，那么最大长度就是 n×1048576（n 的最大值为 2048）。如果指定了 nG，那么最大长度就是 n×1073741824（n 的最大长度是 2）。CLOB 用于存储大 SBCS 或基于字符的混合（SBCS 和 MBCS）数据。因此，CLOB 与 SBCS 或混合代码页关联。

11. 大对象（LOB）和二进制大对象（BLOB）

大对象和类属首字母缩略词 LOB 指的是 BLOB、CLOB 或 DBCLOB 数据类型。LOB 值应遵守适用于 LONG VARCHAR 值的那些限制。即使 LOB 字符串的长度属性为 254 个字节或更少，但这些限制也适用。此数据类型用于存储长度变化的二进制字符串数据。它经常用于存储非传统数据，如文档、图形图像、图片、音频和视频等，还可以容纳用户定义的类型及用户定义的函数的结构化数据。

二进制大对象，BLOB：二进制大对象是变长字符串，最长可达 2147483647 字节。如果只指定了 n，那么 n 就是最大长度。如果指定了 nK，那么最大长度就是 n×1024（n 的最大值为 21097152）。如果指定了 nM，那么最大长度就是 n×1048576（n 的最大值为 2048）。如果指定了 nG，那么最大长度就是 n×1073471824（n 的最大值是 2）。

注意

SQL 不能处理二进制大对象数据。例如不能对二进制大对象进行排序。

12. 图形字符串

图形字符串是表示双字节字符数据的字节序列。图形字符串包括类型为 GRAPHIC（n）的定长图形字符串和类型为 VARGRAPHIC（n）、LONG VARGRAPHIC 的变长图形字符串。字符串长度就是序列中双字节字符的数目。

定长图形字符串，GRAPHIC（n）：定长图形字符串的长度介于 1~127 个双字节字符之间。如果没有指定长度，就认为是 1 个双字节字符。

变长图形字符串，VARGRAPHIC（n）：VARGRAPHIC（n）类型的字符串是变长图形字符串，最大长度可达 16336 个双字节字符。

LONG VARGRAPHIC：LONG VARGRAPHIC 类型的字符串是变长图形字符串，最大长度可达 16350 个双字节字符。

13. 兼容 Unicode 数据

支持的所有数据类型在 Unicode 数据库中也受支持。特别是 Unicode 数据库支持图形字符串数据，并用 UCS-2 编码存储。每个客户机（包括 SBCS 客户机）与 Unicode 数据库连接时，可使用 UCS-2 编码的图形字符串数据类型。

14. 日期和时间数据

日期数据类型用于存储 DATE、TIME 和 TIMESTAMP。日期时间值可在某些算术和字符串操作中使用，而且兼容某些字符串，但是它们既不是字符串，也不是数字。

DATE 是一个由 3 部分组成的值（年、月、日）。年份的取值范围是 0001~9999。月部分的取值范围是 1 到 12。日部分的取值范围是从 1 到 n，其中 n 的值取决于月份。DATE 列长 10 个字节。

TIME 是一个由 3 部分组成的值（小时、分钟、秒）。小时部分的取值范围是 0 到 24，分钟和秒部分的取值范围都是从 0 到 59。如果小时为 24，分钟和秒的值都是 0。TIME 列长 8 个字节。

TIMESTAMP 是一个由 7 部分组成的值（年、月、日、小时、分钟、秒和微秒）。年份部分的取值范围是从 0001~9999。月份部分的取值范围是从 1 到 12。日部分的取值范围是从 1 到 n，其中 n 的值取决于月份。小时部分的取值范围是 0 到 24。分钟和秒部分的取值范围都是 0 到 59。微秒部分的取值范围是从 000000 到 999999。如果小时是 24，那么分钟值、秒的值和微秒的值都是 0。TIMESTAMP 列长 26 个字节。日期时间值的字符串表示：尽管 DATE、TIME 和 TIMESTAMP 的值的内部表示对用户是透明的，日期、时间和时间戳记也可以用字符串来表示，CHAR 标量函数可以用于创建日期时间值的字符串表示。

日期值的字符串表示是一个以数字开始，长度不少于 8 个字符的字符串。日期值的月份和日期中前面的零可以省略。

时间值的字符串表示是以数字开头，长度不少于 4 个字符的字符串。时间值的小时部分前面的零可以省略，秒部分可以完全省略。如果秒的值没有指定，那么就认为是 0。

时间戳记 TIMESTAMP 用于表中声明日期和时间的组合。时间戳记值的字符串表示是以数字开头，长度不少于 16 个字符的字符串。完整的时间戳记字符串表示形式为 yyyy-mm-dd-hh.mm.sss.nnnnnn。时间戳记值的月、日或小时等几部分前面的零可以省略，微秒可以截断或完全省略。如果任何时间戳记值的微秒部分尾零被省略掉了，那么将假定空缺的数位上是零。

15. 数字数据

数字包括符号和精度。如果某个数的值为零，那么认为符号是正号。精度是不包括符号在内的位数。

16. 货币数据

货币数据是一种十进制浮点数据类型，在处理准确的十进制值的业务应用程序中很有用。为十进制数据提供二进制近似值的二进制浮点数据类型不适合在这种应用程序中使用。DECFLOAT 不仅具有 DECIMAL 的准确性，还具有 FLOAT 的一些性能优点，这对于处理货币值的应用程序很有用。

17. XML 数据

XML 数据类型用于定义表中存储 XML 值的列，这些列中存储的所有 XML 值必须是结构良好的 XML 文档。引入 XML 数据类型能够将结构良好的一个 XML 文档以其分层格式存储在表中相应的列中。

5.2.2　创建表

在创建表之前首先要设计表，即确定表的名字、所包含的列名、列的数据类型、长度、是否可为空值、默认值情况、是否要使用以及何时使用约束、默认设置或规则以及所需的索引、哪些列是主键、

哪些列是外键等属性，这些属性就构成了表的结构。

空值通常表示未知、不可用或将在以后添加的数据。若一个列允许为空值，则向表中输入记录值时可不为该列给出具体值；而一个列若不允许为空值，则在输入时必须给出具体值。

下面用两种方式在 MRKJ 数据库中创建学生表，如表 5.3 所示。

表 5.3　学生表

列　名	数 据 类 型	长　度	可　空	默 认 值	说　明
学号	定长字符串型（CHARACTER）	10	NO		主键
姓名	定长字符串型（CHARACTER）	16	NO		
专业	定长字符串型（CHARACTER）	50	YES		
性别	定长字符串型（CHARACTER）	4	YES		值约束：男或女
出生日期	日期型（DATE）		YES		
总分数	小整数型（SMALLINT）		YES	0	
照片	BLOB	2M	YES		
联系方式	XML		YES		
备注	不定长字符串型（VARCHAR）	400	YES		

注意

数据类型 CHARACTER 可简写成 CHAR，以后不再赘述。

1. 用界面方式创建表

（1）在“控制中心”窗口的对象视图中，选择 MRKJ 数据库下的“表”文件夹，然后单击鼠标右键，在弹出的快捷菜单中选择“创建”命令，启动“创建表向导”对话框，如图 5.8 所示。

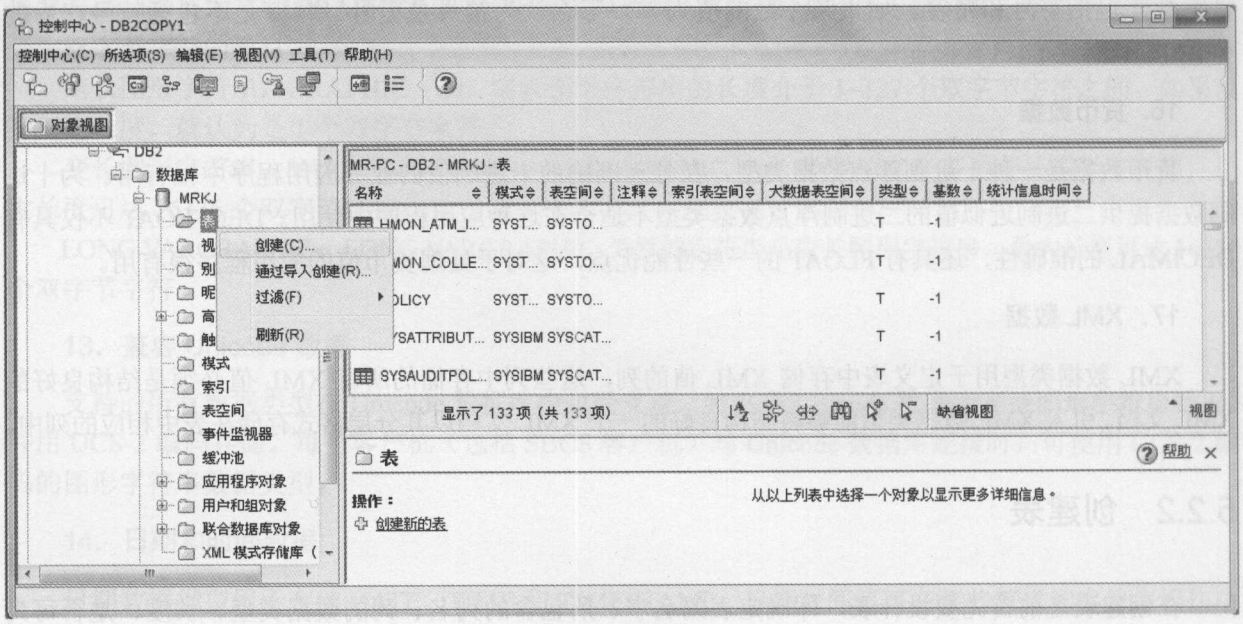


图 5.8　创建表

（2）在“创建表向导”对话框的“标识新表的模式和名称”界面中，在“表模式”下拉列表框中默认的表模式是当前登录数据库用户 ID，也可以在“表模式”下拉列表框中输入自己想要的表模式，如果数据库中不存在该表模式，则数据库自动创建一个新的表模式，这里选择默认的“MR”。在“表名”文本框中输入表名“学生表”，在“注释”文本框中输入“学生信息表”，如图 5.9 所示。然后单击“下一步”按钮。

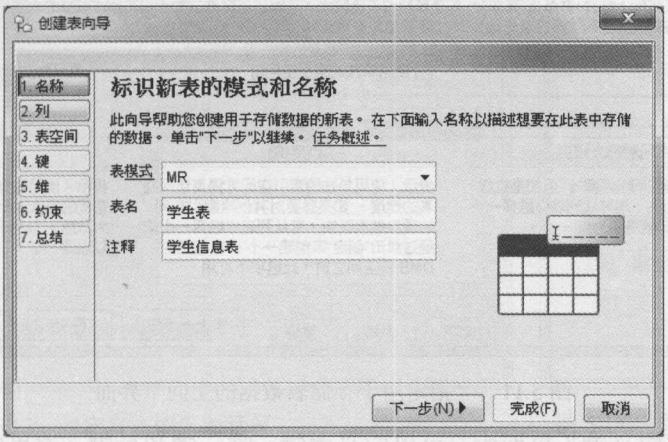


图 5.9　创建表模式、表名和注释

（3）在“更改新表的列定义”界面中，单击“添加”按钮从而启动“添加列”对话框，逐个添加列。根据表结构设计，输入列名、选择数据类型、需要时输入长度、设定默认值和选择是否可为空。添加完后如图 5.10 所示。单击“下一步”按钮。

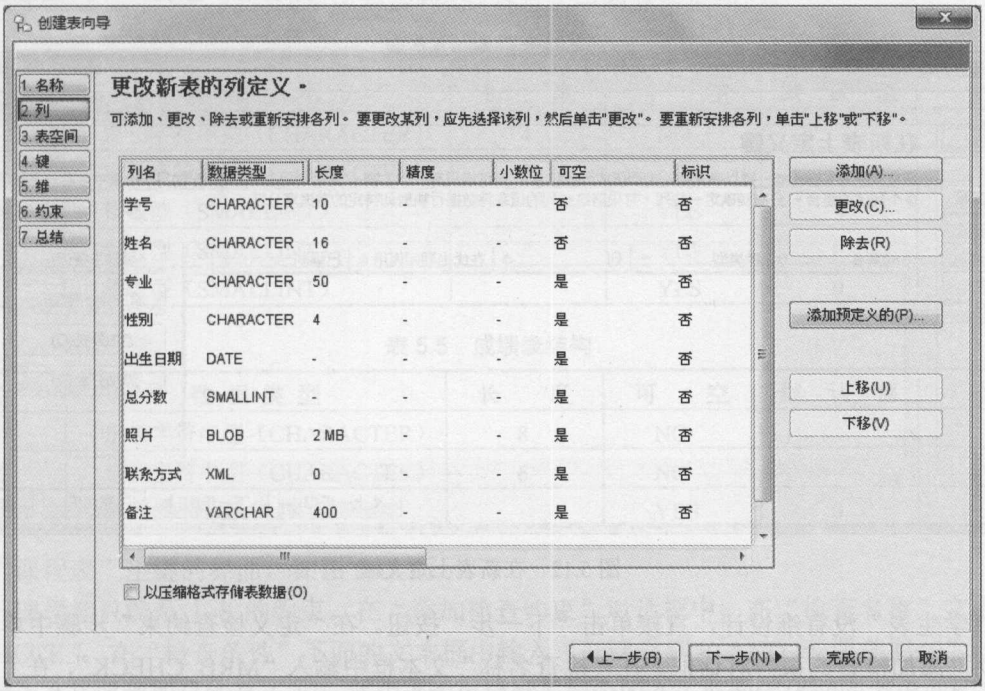


图 5.10　“更改新表的列定义”界面

（4）在“指定用于存储表数据的空间”界面中，在“表空间”下面的列表框中选择前面创建的表空间“CUSTOMDATASPACE1”，如图 5.11 所示。单击“下一步”按钮。

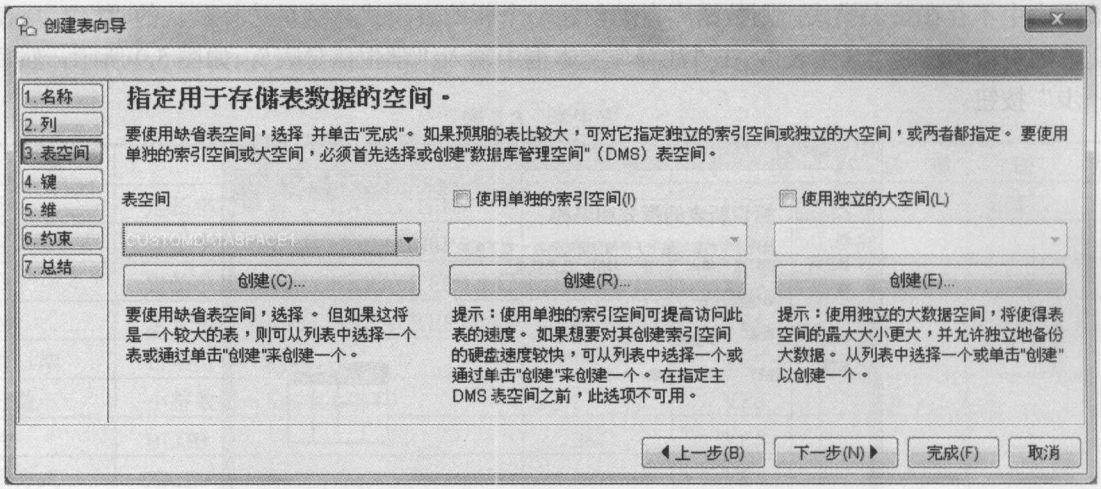


图 5.11　“指定用于存储表数据的空间”界面

（5）在“在新表上定义键”界面中，可以定义主键、唯一键和外键。这里需要将学生表的“学号”列添加为主键。单击“添加主键”按钮，如图 5.12 所示，启动“定义主键”对话框。在“主键名”文本框中输入“MRB_PRIMARY_KEY”。然后选中列表中的“学号”，单击“>”按钮，可以看到左边列表中的“学号”已经移至右边了，如果有多个列组合成主键，重复操作刚才步骤即可，如图 5.13 所示。然后单击“确定”按钮，这时回到如图 5.12 所示的对话框，单击“下一步”按钮，进入到“维设计”对话框。

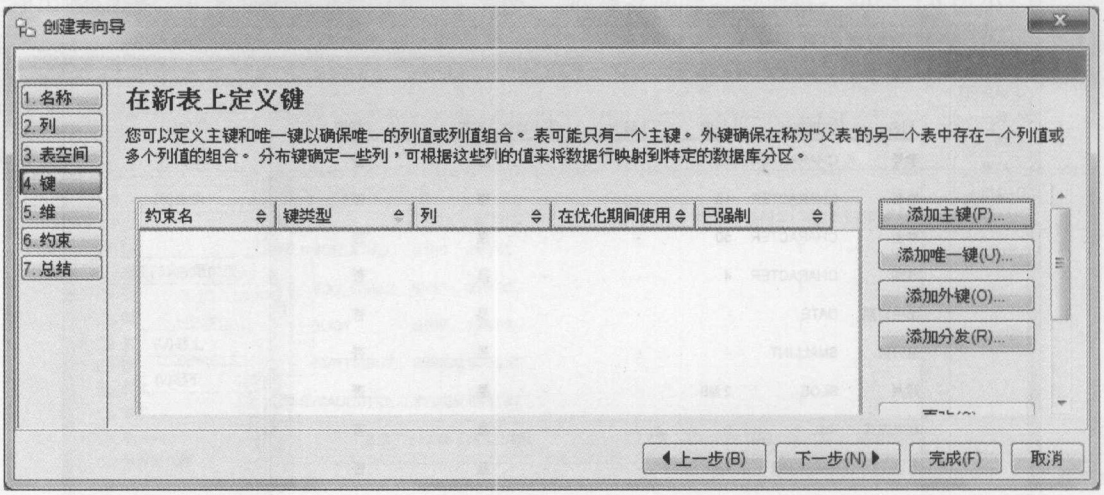


图 5.12　在新表上定义键

（6）“学生表”没有维设计，直接单击“下一步”按钮。在“定义检查约束”步骤中单击“添加”按钮，启动“添加检查约束”对话框。在“检查名称”文本框中输入“MRB_CHECK”，在“检查条件”文本框中输入“性别='男' OR 性别='女'”，在“注释”文本框中输入“性别只能为男或女”。选中“约束

属性”栏中的“在查询优化期间考虑此约束”和“强制此约束”复选框，如图 5.14 所示。单击“确定”按钮回到“定义检查约束”对话框，在该对话框中单击“完成”按钮，“学生表”创建完成。

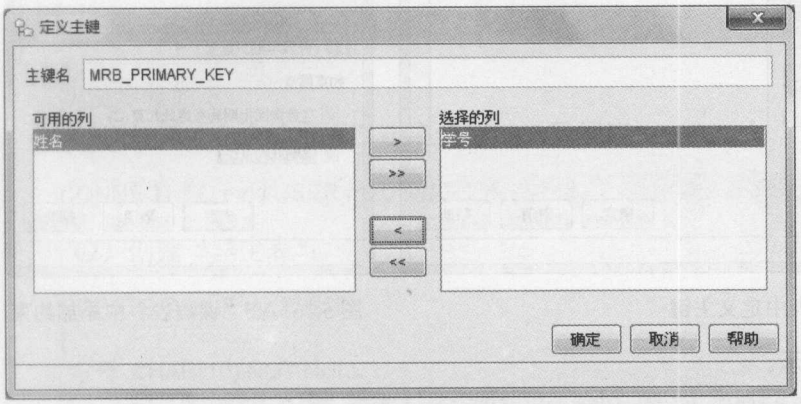


图 5.13　“定义主键”对话框

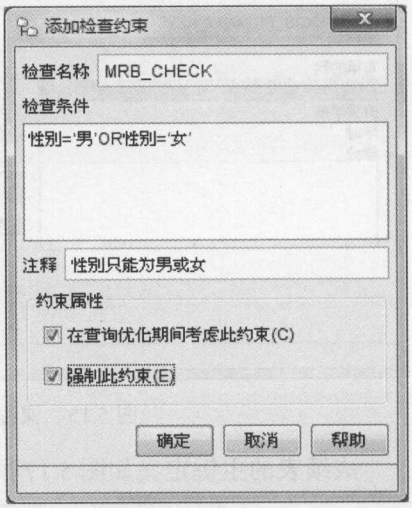


图 5.14　“添加检查约束”对话框

注意

性别的检查约束中，男或女用英文状态下的单引号引上，否则会出现错误。

下面通过控制中心创建“课程表”和“成绩表”，表结构如表 5.4 和表 5.5 所示。

表 5.4　课程表结构

列　名	数 据 类 型	长　度	可　空	默 认 值	说　明
课程号	定长字符串型（CHARACTER）	4	NO		主键
课程名	定长字符串型（CHARACTER）	50	NO		
开课学期	整数型（SMALLINT）		YES		只能为 1~8
学时	整数型（SMALLINT）		YES	0	
学分	整数型（SMALLINT）		YES	0	

表 5.5　成绩表结构

列　名	数 据 类 型	长　度	可　空	默 认 值	说　明
学号	定长字符串型（CHARACTER）	8	NO		主键
课程名	定长字符串型（CHARACTER）	6	NO		主键
成绩	整数型（SMALLINT）		YES	0	

创建“课程表”主键的界面，如图 5.15 所示。

定义开课学期只能是 1~8 的约束。在“添加检查约束”对话框中，在“检查名称”文本框中输入“KCB_CHECK”，在“检查条件”下面的文本框中输入“开课学期>0 AND 开课学期<9”，在“注释”文本框中输入“开课学期只能是 1~8”。选中“约束属性”栏中的“在查询优化期间考虑此约束”和“强制此约束”复选框，如图 5.16 所示。

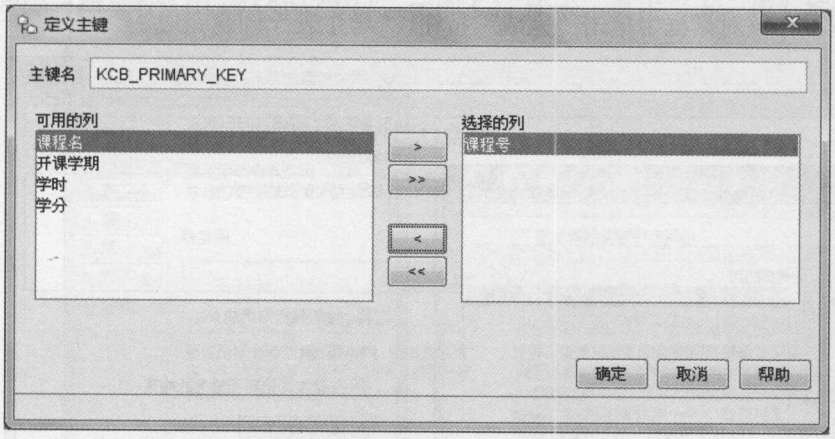


图 5.15　课程表中定义主键

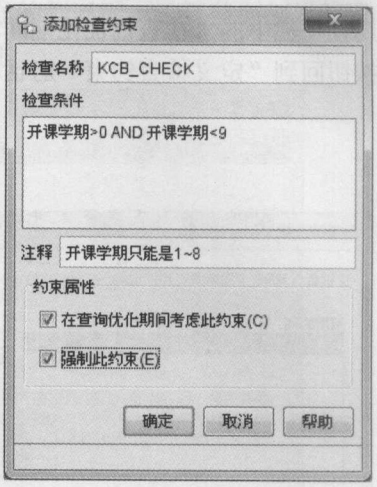


图 5.16　在“课程表”中添加约束

成绩表的主键定义如图 5.17 所示。

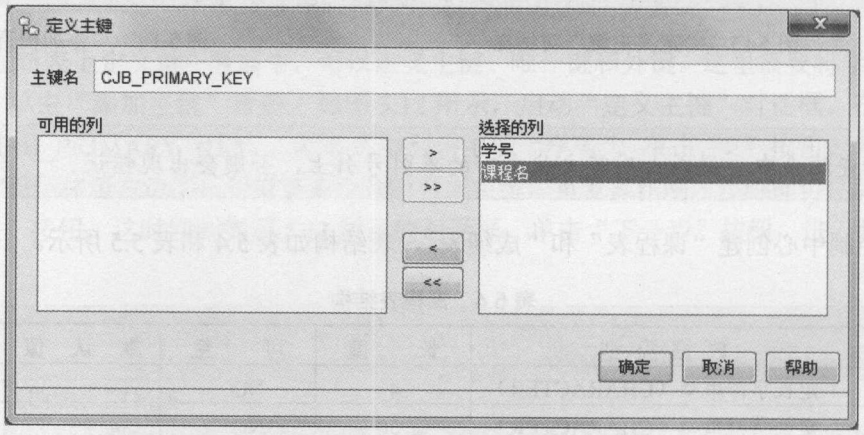


图 5.17　定义成绩表主键

2. 用命令方式创建表

使用 CREATE TABLE 语句来创建表，完整的 CREATE TABLE 语句的语法非常复杂，可以使用复杂语句来定义表的所有属性。但是，如果所有属性都使用默认值，那么创建表的语句就非常简单了。

使用默认值创建表的语法为：

```
CREATE TABLE tableName({column_name data_type[column_options]}[,...n])
```

参数说明：

- ☑ [,...n]表示前面{}中的内容可以有多组，每组之间用逗号隔开。
- ☑ table_name 不一定包含限定词。表名对系统目录中的所有表、视图和别名来说必须唯一。而且名称不能是 SYSIBM、SYSCAT、SYSFUN、SYSSTAT。
- ☑ column_name 用于对表中的列命名。并且在该表其他列中必须唯一。对列存在的任何 column_options 进一步定义该列的属性。选项包括用于防止列包含控制的 NOT NULL、用于

LOB 数据类型的特定选项、引用类型列的 SCOPE、对表的任何约束以及列的任何默认值。

【例 5.1】　使用命令在 MRKJ 数据库创建“学生表”，表结构如表 5.3 所示。（**实例位置：光盘\TM\sl\5\1**）

注意

如果已经创建了某个表，则不能再创建同名表。也就是说，如果已经用界面方式创建了“学生表”，则不能用命令方式再次创建“学生表”。

实现步骤如下。

（1）打开“命令编辑器”连接到 MRKJ 数据库：

```
CONNECT TO mrkj USER mr USING 1;
```

（2）创建“学生表”：

```
CREATE TABLE mr.学生表
(
 学号 CHAR(10) NOT NULL,
      --指定表中列的格式：列名 数据类型[长度][是否可空]
      --其中数据类型 CHARACTER 可简写成 CHAR
姓名 CHAR (16) NOT NULL,
专业 CHAR (50),
性别 CHAR (4) NOT NULL,
出生日期 DATE,
总分数 SMALLINT WITH DEFAULT 0,
备注 VARCHAR(400),
联系方式 XML,
照片 BLOB (2M) LOGGED NOT COMPACT,
      --LOGGED 表示已记录,NOT COMPACT 表示非压缩
CONSTRAINT xsb_primary_key PRIMARY KEY(学号),  --定义主键
CONSTRAINT xsb_check CHECK (性别='男' OR 性别='女')
      --“性别”列的值只能是“男”或“女”
      ENFORCED        --强制此约束
      ENABLE QUERY OPTIMIZATION   --在查询优化期间考虑此约束
)IN customdataspace1;    --学生表创建在表空间 customdataspace1 上
```

（3）添加注释：

```
COMMENT ON TABLE MR.学生表 IS'学生情况表';  --学生表的注释
```

【例 5.2】　在 MRKJ 数据库中创建“课程表”，表结构如表 5.4 所示（**实例位置：光盘\TM\sl\5\2**）。

实现步骤如下。

（1）连接 MRKJ 数据库：

```
CONNECT TO mrkj USER mr USING 1;
```

（2）创建“课程表”：

```
CREATE TABLE mr.课程表
(
```

```
    课程号 CHAR(4) NOT NULL,
    课程名 CHAR(50) NOT NULL,
    开课学期 SMALLINT,
    学时 SMALLINT,
    学分 SMALLINT,
    CONSTRAINT kcb_primary_key PRIMARY KEY(课程号),  --定义主键
    CONSTRAINT kcb_check
    CHECK(开课学期>0 AND 开课学期<9)  --定义检查约束
    ENFORCED ENABLE QUERY OPTIMIZATION
)IN customdataspace1;
```

（3）为“课程表”以及表上的约束添加注释：

```
COMMENT ON TABLE mr.课程表 IS '课程信息表';
COMMENT ON CONSTRAINT mr.课程表.kcb_check IS '开课学期只能为 1-8';
    --课程表开课学期列上的检查约束的注释
```

【例 5.3】 使用命令创建“成绩表”，表结构如表 5.5 所示（**实例位置：光盘\TM\sl\5\3**）。实现步骤如下。

（1）连接 MRKJ 数据库：

```
CONNECT TO mrkj USER mr USING 1;
```

（2）创建“成绩表”：

```
CREATE TABLE mr.成绩表
(
    学号 CHAR(8) NOT NULL,
    课程号 CHAR(6) NOT NULL,
    成绩 SMALLINT NOT NULL WITH DEFAULT 0,
    CONSTRAINT cjb_primary_key PRIMARY KEY(学号,课程号)
)IN customdataspace1;
```

5.2.3 修改表

在创建表之后，使用过程中可能对表结构、约束或其他列的属性进行修改。通过“控制中心”可以完成对表的修改。一个已经存在的表可以进行的修改操作包括：

- ☑ 更改表名。
- ☑ 增加列。
- ☑ 删除列。
- ☑ 修改已有列的属性（列名、数据类型、是否为空值）。

1. 用界面的方式修改表

更改表名的方法以及步骤如下。

DB2 中允许改变表的名字，但当表名改变后，与此相关的某些对象如视图，以及通过表名与表相

关的存储过程将无效。因此，建议一般不要更改一个已有的表名，特别是在其上定义了视图或建立了相关的表。

使用“控制中心”将“成绩表”改名为“score”。实现步骤如下：

（1）在“控制中心”窗口的对象视图中选择 MRKJ 数据库下的“表”文件夹，在内容窗格中找到“成绩表”，然后单击鼠标右键，在弹出的快捷菜单中选择“重命名”命令，从而启动“重命名表”对话框，如图 5.18 所示。

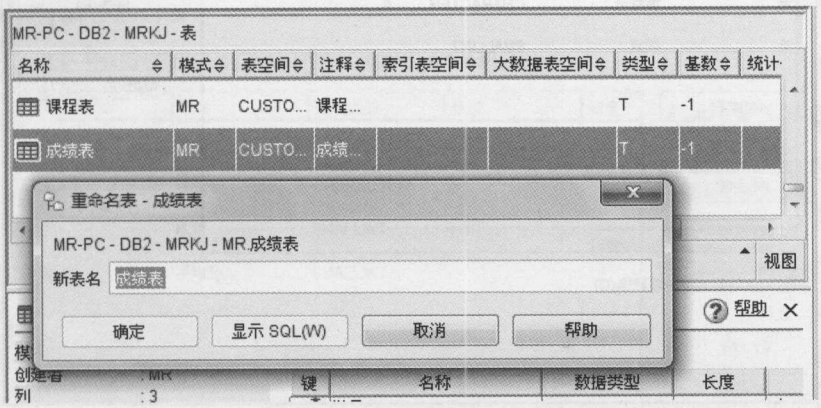


图 5.18　重命名成绩表

（2）在“重命名表”对话框中，在“新表名”文本框中输入“score”，然后单击“确定”按钮，如图 5.19 所示。

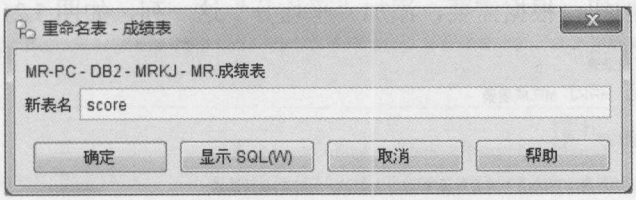


图 5.19　重命名成绩表

注意

不管输入的字母是大写还是小写，DB2 一律更改为大写字母。

说明

根据本书需要，请读者把表“score”改回“成绩表”。

向“成绩表”中添加“平均分”列，该列类型为 SMALLINT。实现步骤如下：

（1）在“控制中心”窗口的对象视图中找到 MRKJ 数据库下的“表”文件夹，在内容窗格中找到成绩表，然后单击鼠标右键，在弹出的快捷菜单中选择“改变”命令，从而启动“改变表”对话框，如图 5.20 所示。

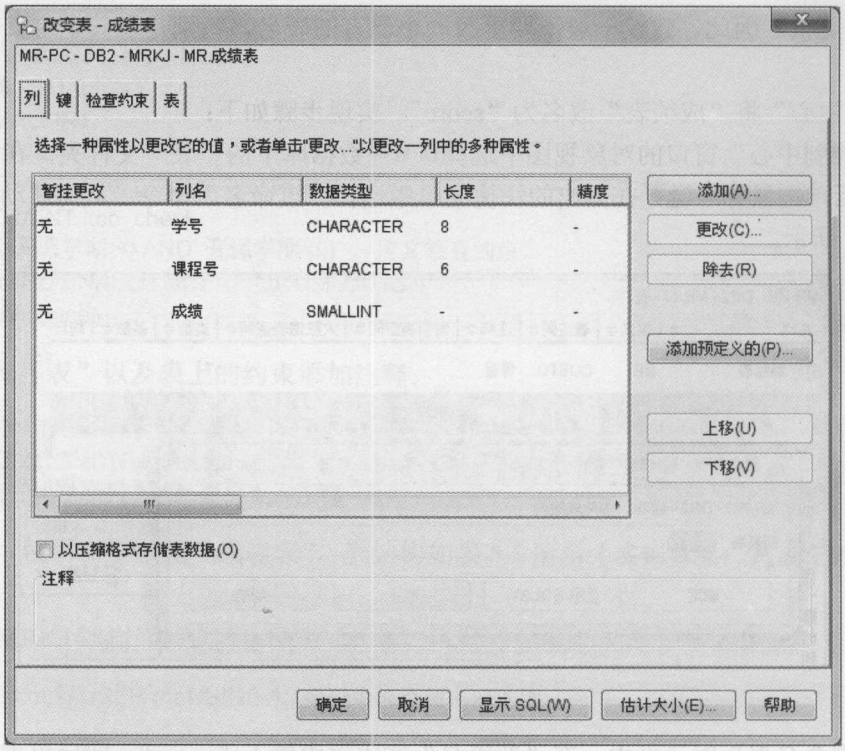


图 5.20　“改变表”对话框

（2）单击“添加”按钮，根据需要，添加“平均分”这一列，如图 5.21 所示。

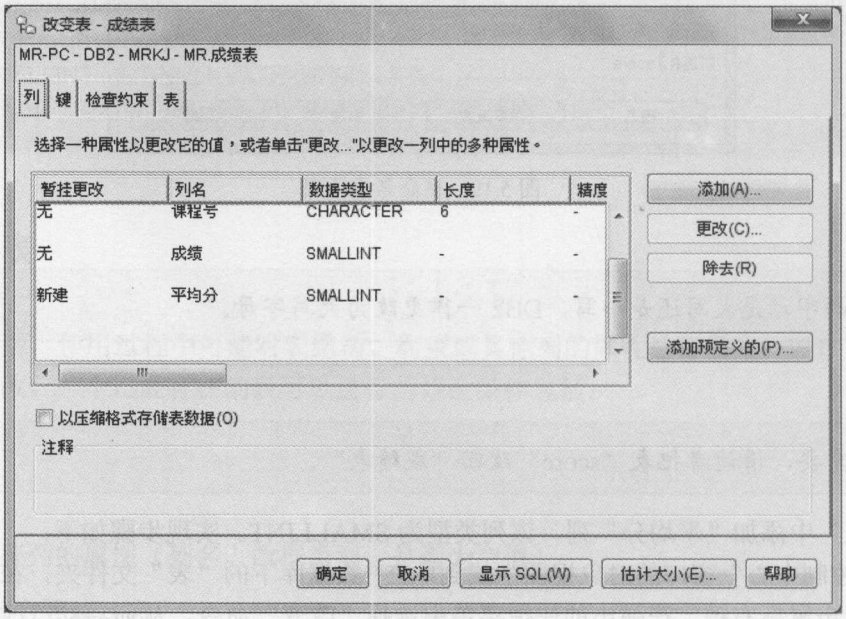


图 5.21　在成绩表中添加“平均分”列

删除“成绩表”中的“平均分”列，实现步骤如下。

（1）在“控制中心”窗口的对象视图选择 MRKJ 数据库下的“表”文件夹，在内容窗格中找到成绩表，单击鼠标右键，在弹出的快捷菜单中选择“改变”命令，则启动“改变表”对话框，如图 5.22 所示，并弹出一个“DB2 消息”对话框，用来提示用户在进行更改操作时需要花费一些时间，如图 5.23 所示。

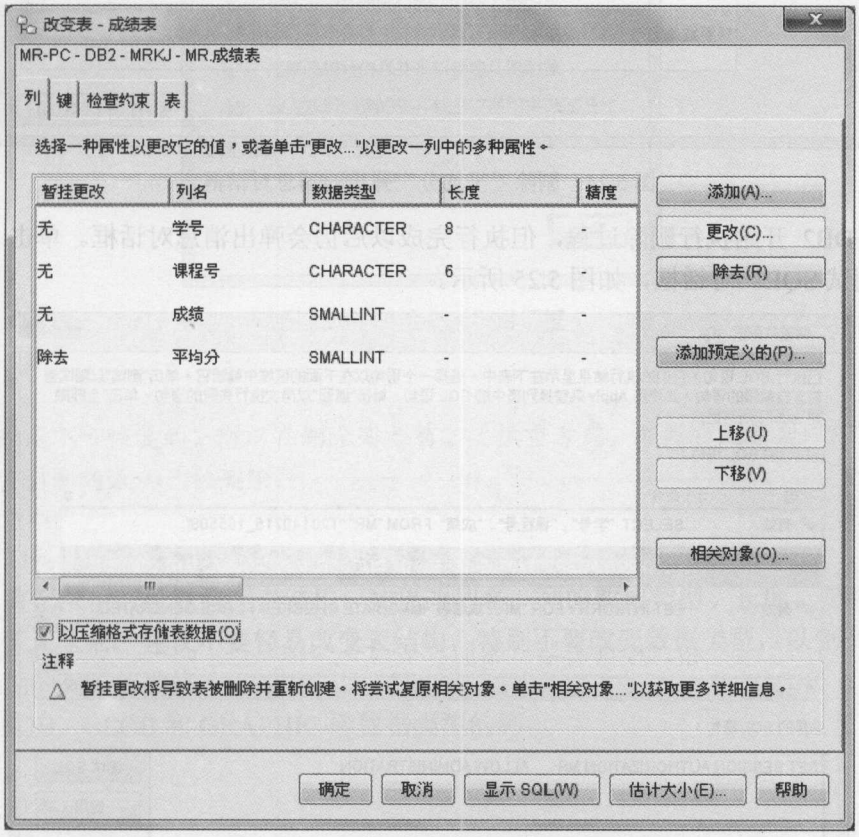


图 5.22　在“成绩表”中删除“平均分”列

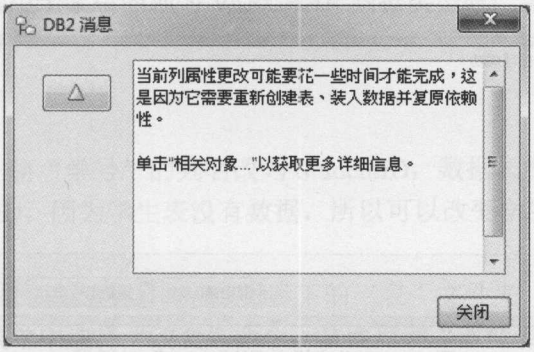


图 5.23　DB2 消息对话框

（2）此时再次弹出“DB2 消息”对话框用来警示用户，防止用户因误操作而带来不可估量的损失。消息对话框提示用户是否想要继续。单击“是”按钮继续删除，如图 5.24 所示。

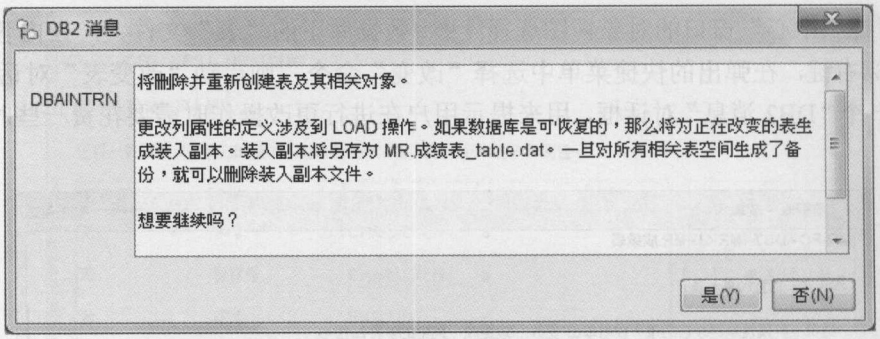


图 5.24　删除“平均分”弹出的消息对话框

（3）此时 DB2 开始执行删除过程，但执行完成以后仍会弹出消息对话框。单击“确定”按钮可打开“编辑并测试 SQL”对话框，如图 5.25 所示。

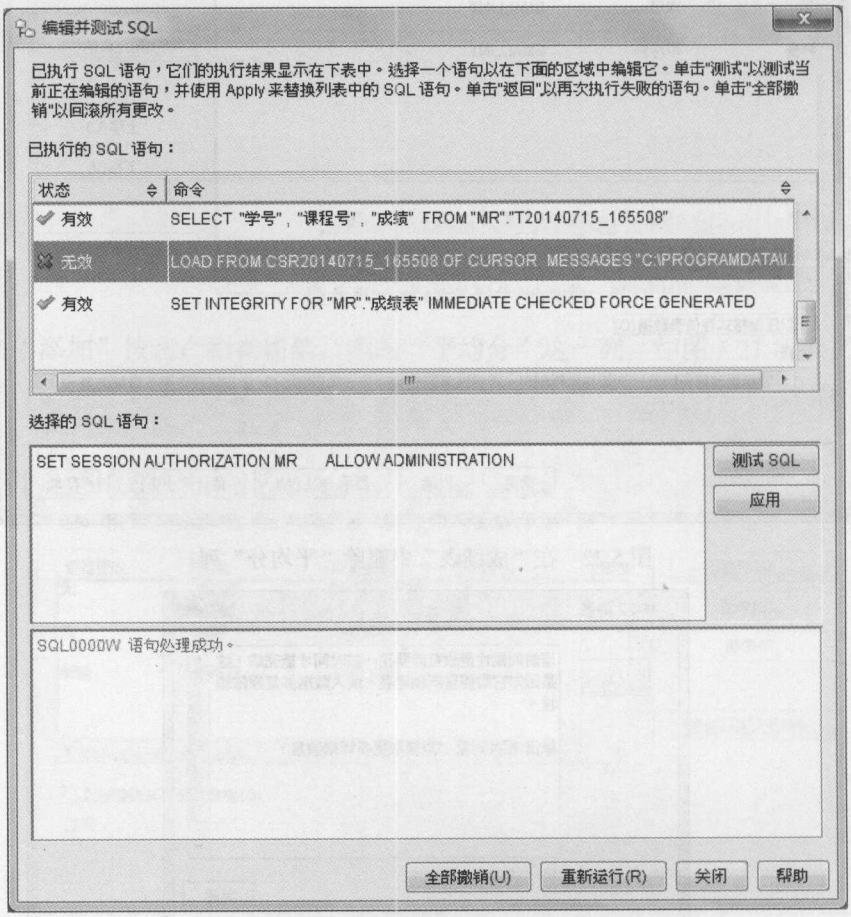


图 5.25　“编辑并测试 SQL”对话框

（4）“编辑并测试 SQL”对话框中列出了详细的错误报告，这里提示的错误是一个非关键性错误，单击“关闭”按钮忽略此错误，如图 5.26 所示。单击“关闭”按钮继续忽略此错误。

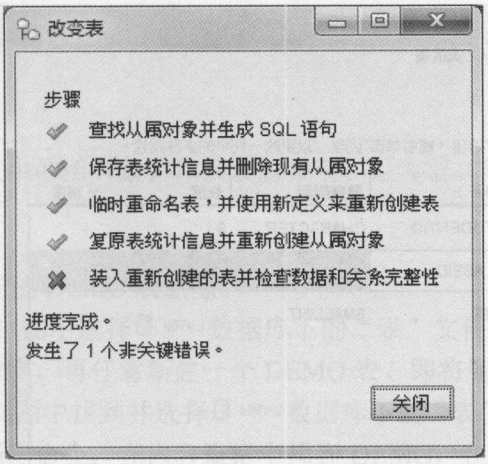


图 5.26　“改变表”对话框

注意

被删除的列是不可恢复的，所以在删除列之前需要慎重考虑。并且在删除列以前，必须保证基于该列的所有索引和约束都已被删除。

修改已有列的属性的方法和步骤如下：

在表中尚未有记录值，可以修改表结构，如更改列名、列的数据类型、长度、是否允许为空等属性。但当表中有了记录后，建议不要轻易改变表结构，特别不要改变数据类型，以免产生错误。

具有下列特性的列不能被修改：

- ☑ 具有 CLOB、BLOB 和 GRAPHIC 等数据类型的列。
- ☑ 计算列。
- ☑ 全局标识符列。
- ☑ 复制列。
- ☑ 用于索引的列（但若用于索引的列为 VARCHAR 数据类型时，可以增加列的长度）。
- ☑ 用于主键或外键约束。
- ☑ 用于约束的列。
- ☑ 关联有默认值的列。

在创建的“成绩表”中，将“学号”的列名改为 StudentID，数据长度由 8 改为 10，允许为空值；将“课程号”列名改为 ClassID。因为学生表没有数据，所以可以改变学生表的结构。

实现步骤如下：

（1）在“控制中心”对象视图选择 MRKJ 数据库下的“表”文件夹，在内容窗格找到成绩表，单击鼠标右键，在弹出的快捷菜单中选择“改变”命令，从而启动“改变表”对话框。

（2）在“改变表”对话框的“列”选项卡中修改相应的列，将列名“学号”修改为 StudentID，将“课程号”修改为 ClassID，修改后如图 5.27 所示（注意：DB2 执行过程中列名中的字母如果是小写也会被自动改为大写）。单击“确定”按钮，弹出“DB2 消息”对话框。在每个“DB2 消息”对话框中，单击“是”按钮或“关闭”按钮，和删除列一样，DB2 会进行一个负责的操作，最后修改完成。

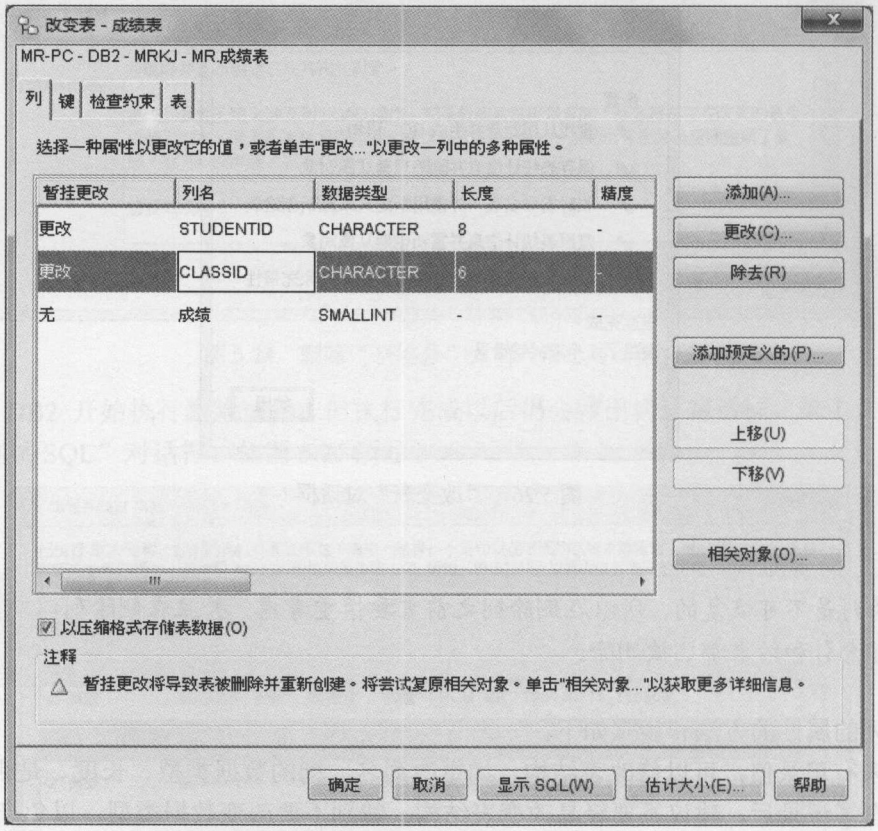


图 5.27　修改“成绩表”的列属性

由于本书讲解的需要，请读者在按照上述过程操作完成后将“成绩表”中的“学号”和“成绩”列的名称仍恢复到原来的状态。

2. 用命令方式修改表

使用命令向“成绩表”中添加“平均分”列，该列类型为 SMALLINT，不允许为空。

注意

如果已经存在“平均分”列，则不能再创建同名列。

实现步骤如下。

（1）连接到 MRKJ 数据库：

```
CONNECT TO mrkj USER mr USING 1;
```

（2）给“成绩表”增加“平均分”列：

```
ALTER TABLE MR.成绩表                          --要被修改的目的表
ADD COLUMN  平均分  SMALLINT NOT NULL;  --增加列
```

5.2.4　删除表

当删除表时，表的定义、表中的所有数据以及表的索引、触发器、约束等均被删除。

1. 用界面方式删除表

将 MRKJ 数据库中任意新建的 DEMO 表删除。

（1）在“控制中心”对象视图中选择 MRKJ 数据库下的“表”文件夹，然后单击鼠标右键，在弹出的快捷菜单中选择“创建”命令，再任意新建一个 DEMO 表。即将删除的就是这个表。

（2）在“控制中心”对象视图中找到并选择 MRKJ 数据库下的“表”文件夹，然后单击鼠标右键，在弹出的快捷菜单中选择“刷新”命令，在内容窗格中找到 DEMO，在右键快捷菜单中选择“删除”命令，如图 5.28 所示，弹出删除表“确认”对话框。在删除表“确认”对话框确认无误后，单击“确定”按钮，DEMO 表即被删除，如图 5.29 所示。

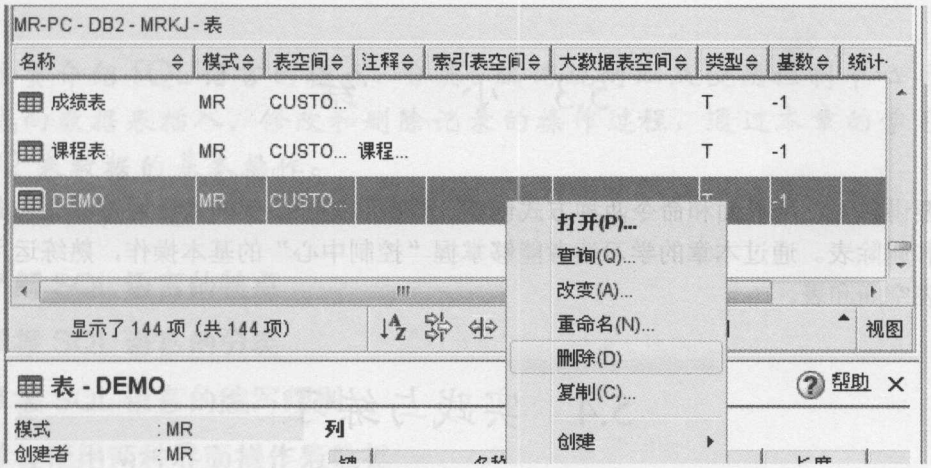


图 5.28　删除 DEMO 表

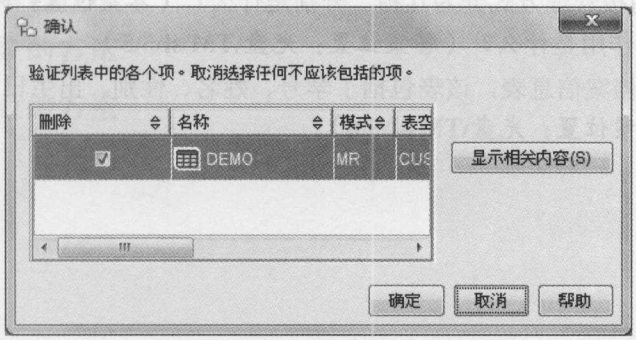


图 5.29　确认删除 DEMO 表

2. 用命令方式删除表

删除表需要使用 DROP TABLE 命令。当删除一个表时，也会删除 SYSCAT.TABLES 系统目录中包

含有关该表的信息的那一行，并会影响从属于该表的任何其他对象。例如，会删除所有的列名、基于该表的任何列创建的索引、该表为父表或从属表的所有引用约束；该表的所有视图、从属于该表的所有程序包和高速缓存的动态 SQL 和 XQuery 语句、从属于该表的所有触发器会标记为不可用。对删除的表和从属视图的所有特权被隐式撤销。

DROP TABLE 语句语法为：

```
DROP TABLE table_name;
```

实现步骤如下。

（1）连接 MRKJ 数据库：

```
CONNECT TO mrkj USER mr USING 1;
```

（2）删除 DEMO 表：

```
DROP TABLE mr.demo;
```

5.3 小　　结

本章主要讲解了使用界面和命令两种方式创建、修改和删除表空间以及通过界面和命令两种方式创建、修改和删除表。通过本章的学习读者能够掌握“控制中心”的基本操作，熟练运用界面和代码管理器管理表空间和表。

5.4 实践与练习

（1）DB2 的表空间按管理方式分为几种，分别是什么？（**答案位置：光盘\TM\sl\5\4**）

（2）DB2 表空间的作用是什么？（**答案位置：光盘\TM\sl\5\5**）

（3）创建一个学生档案信息表，该表包括了学号、姓名、性别、出生日期、系别编号、班级编号和建档日期等信息。（**答案位置：光盘\TM\sl\5\6**）

第 6 章

表数据的基本操作

（视频讲解：30 分钟）

本章主要介绍 SQL 语言的特点、分类、编写规则以及使用控制中心与命令编辑器两种方式向数据表插入、修改和删除记录的操作过程，通过本章的学习，读者可以熟悉 DB2 表数据的基本操作。

通过阅读本章，您可以：

- 了解 SQL 语言的特点
- 掌握 SQL 语言的分类
- 掌握 SQL 语言的编写规则
- 掌握使用两种界面操作表数据
- 掌握插入记录的方法
- 掌握修改记录的方法
- 掌握删除记录的方法

6.1 SQL 语言的概述

SQL 全称是结构化查询语言，英文译作 Structured Query Language，它是一种在关系型数据库中定义和操纵数据的标准语言。最早是由 IBM 的圣约瑟研究实验室为其关系数据库管理系统 SYSTEM R 开发的一种查询语言，当时称为 SEQUEL2，也就是目前的 SQL 语言。

1976 年 Oracle 公司首先提供了商用的 SQL 语言，同年，IBM 公司在 DB2 和 SQL/DS 数据库系统中也采用了 SQL 语言。1986 年 10 月，美国 ANSI（美国国家标准化组织）采用 SQL 作为关系数据库管理系统的标准语言（ANSI X3.1935-1986），后来 SQL 语言被国际标准化组织（ISO）采纳为国际标准。

随着数据库技术的发展，SQL 标准也在不断地进行扩展和修正，数据库标准委员会先后又推出了 SQL-89、SQL-92 及 SQL-99 标准。目前，所有主要关系型数据库管理系统都支持某个标准的 SQL 语言，其中大部分数据库都遵守 ANSI SQL89 标准。

“关系数据库”是表的集合，它们之间的关系由表中数据项的值来描述。SQL 是标准化的语言，也就是说不管目前使用的是哪种关系型数据库管理系统，该语言基本上都是一样的。并且 SQL 是用来定义和操作数据库对象的语言，经常要处理的对象就是数据库表。SQL 语言可用来定义数据库表，往表中插入数据，改变表中的数据，从表中检索数据。

6.1.1 SQL 语言的特点

SQL 语言结构简洁，功能强大，简单易学，自从被国际标准化组织（ISO）采纳为国际标准以后，SQL 语言得到了广泛的应用，它主要有以下特点。

（1）集合性。SQL 可以在高层的数据结构上进行工作，工作时不是单条地处理记录，而对数据进行成组的处理。所有 SQL 语句都接受集合作为输入，并且返回集合作为输出，SQL 的集合特性允许一条 SQL 语句的结果作为另一条 SQL 语句的输入。

（2）统一性。SQL 为许多任务提供了统一的命令，这样方便用户学习和使用，基本的 SQL 命令只需很少时间就能学会，甚至最高级的命令也可以在几天内掌握。数据库的操作任务通常包括以下几方面：

- ☑ 查询数据。
- ☑ 在表中插入、修改和删除记录。
- ☑ 建立、修改和删除数据对象。
- ☑ 控制对数据和数据对象的读写。
- ☑ 保证数据库一致性和完整性。

（3）易于移植性。由于所有主要的关系型数据库管理系统都支持 SQL 语言，这样用户就比较容易将原先使用 SQL 的技能从一个 RDBMS 转到另一个，而且用兼容 SQL 标准编写的程序都是可以移植的。

6.1.2　SQL 语言的分类

SQL 是关系型数据库的基本操作语言，是数据库管理系统与数据库进行交互的接口。它将数据查询、数据操纵、事务控制、数据定义和数据控制等功能集于一身，而这些功能又分别对应着各自的 SQL 语言，具体如下。

- ☑ 数据查询语言（DQL）。用于检索数据库中的数据，主要是 SELECT 语句，它在操作数据库的过程中使用最为频繁。
- ☑ 数据操纵语言（DML）。用于改变数据库中的数据，主要包括 INSERT、UPDATE 和 DELETE 3 条语句。其中，INSERT 语句用于将数据插入到数据库中，UPDATE 语句用于更新数据库中已经存在的数据，而 DELETE 语句则用于删除数据库中已经存在的数据。
- ☑ 事务控制语言（TCL）。用于维护数据的一致性，包括 COMMIT、ROLLBACK 和 SAVEPOINT 3 条语句。其中，COMMIT 语句用于提交对数据库的更改，ROLLBACK 语句用于取消对数据库的更改，而 SAVEPOINT 语句则用于设置保存点。
- ☑ 数据定义语言（DDL）。用于建立、修改和删除数据库对象。例如，可以使用 CREATE TABLE 语句创建表；使用 ALTER TABLE 语句修改表结构；使用 DROP TABLE 语句删除表。
- ☑ 数据控制语言（DCL）。用于设置或者更改数据库用户或角色权限的语句，主要包括 GRANT 和 REVOKE 两条命令。其中，GRANT 命令用于给用户或角色授予权限，而 REVOKE 命令则用于收回用户或角色所具有的权限。

6.1.3　SQL 语言的编写规则

SQL 关键字不区分大小写，既可以使用大写格式，也可以使用小写格式，或者大小写格式混用。

【例 6.1】　编写以下两条语句，对关键字（INSERT、INTO 和 VALUES）分别使用大写格式、小写格式或大小写混用格式，代码如下（**实例位置：光盘\TM\sl\6\1**）。

```
INSERT INTO mr.学生表 (学号,姓名,专业,性别,出生日期,总分数) VALUES ('14004','蒋阳泉','计算机应用技术','男',
'1991-07-13',84);

Insert into mr.学生表 (学号,姓名,专业,性别,出生日期,总分数) values ('14004','蒋阳泉','计算机应用技术','男',
'1991-07-13',84);
```

分别执行上面的 SELECT 语句，会发现结果完全相同。

表名和列名不区分大小写，它们既可以使用大写格式，也可以使用小写格式，或者大小写格式混用。

【例 6.2】　编写以下 3 条语句，对“表名和列名”分别使用大写格式、小写格式或大小写混用格式，代码如下（**实例位置：光盘\TM\sl\6\2**）。

```
--假如 MRKJ 数据库中有个数据表，表名叫 USER。在表中有两列分别是 USERNAME（姓名）和 AGE（年龄）
--这两个列。如果想要往 USER 表中插入数据，命令如下：
```

```
INSERT INTO mr.USER (USERNAME,AGE) VALUES ('wanghaha',23)

INSERT INTO mr.user (USERNAME,AGE) VALUES ('wanghaha',23)

INSERT INTO mr.user (username,age) VALUES ('wanghaha',23)
```

分别执行这 3 条 INSERT 语句，会发现结果完全相同。

6.2 使用界面操作表数据

创建数据库和表后，需要对表中的数据进行操作，操作包括插入、修改和删除。本节将通过以下两种方式操作表中的数据：

☑ 用“控制中心”的“打开表”对话框。

☑ 用“控制中心”的“命令编辑器”选项卡。

1. 用“控制中心”的“打开表”对话框操作表数据

（1）在“控制中心”对象视图中选择 MRKJ 数据库下的“表空间”文件夹，在内容窗格中选择学生表，然后单击鼠标右键，在弹出的快捷菜单中选择“打开”命令，如图 6.1 所示。

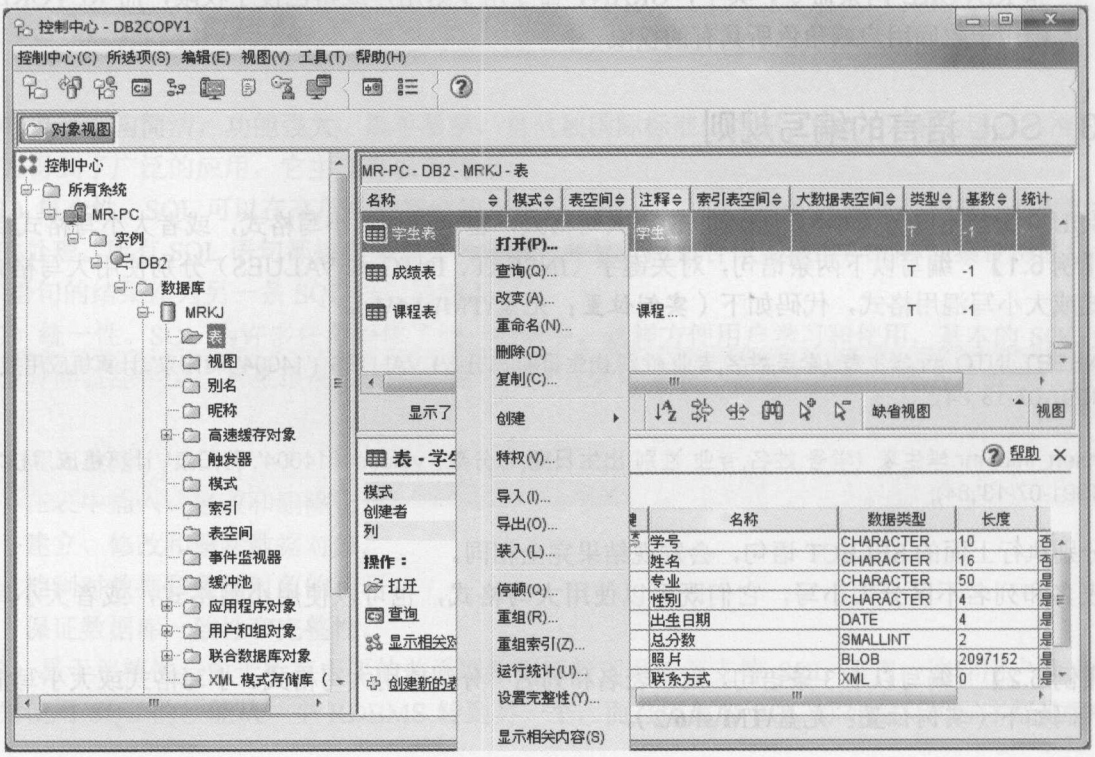


图 6.1 打开“学生表”

（2）在弹出的“打开表”对话框中，“学生表”的信息显示如图 6.2 所示。在此对话框中，可向表中插入记录，也可以删除和修改记录。

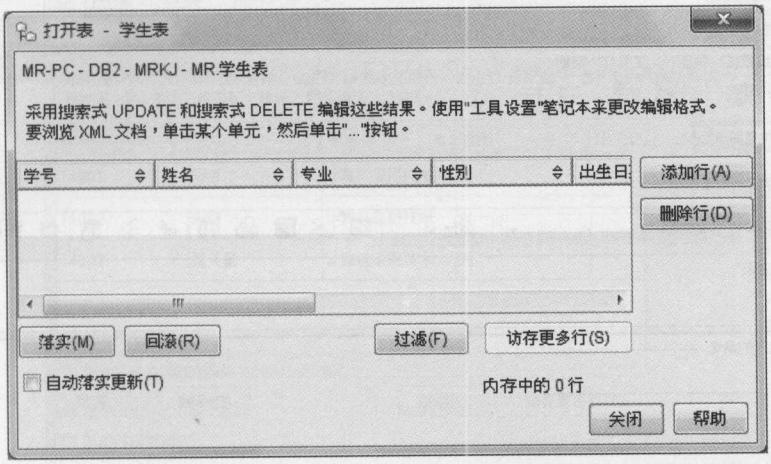


图 6.2　“打开表”对话框

2. 用“控制中心”的“命令编辑器”选项卡操作表数据

（1）在“控制中心”对象视图中选择 MRKJ 数据库下的“表空间”文件夹，内容窗格中找到学生表，单击鼠标右键，在弹出的快捷菜单中选择“查询”命令，如图 6.3 所示。

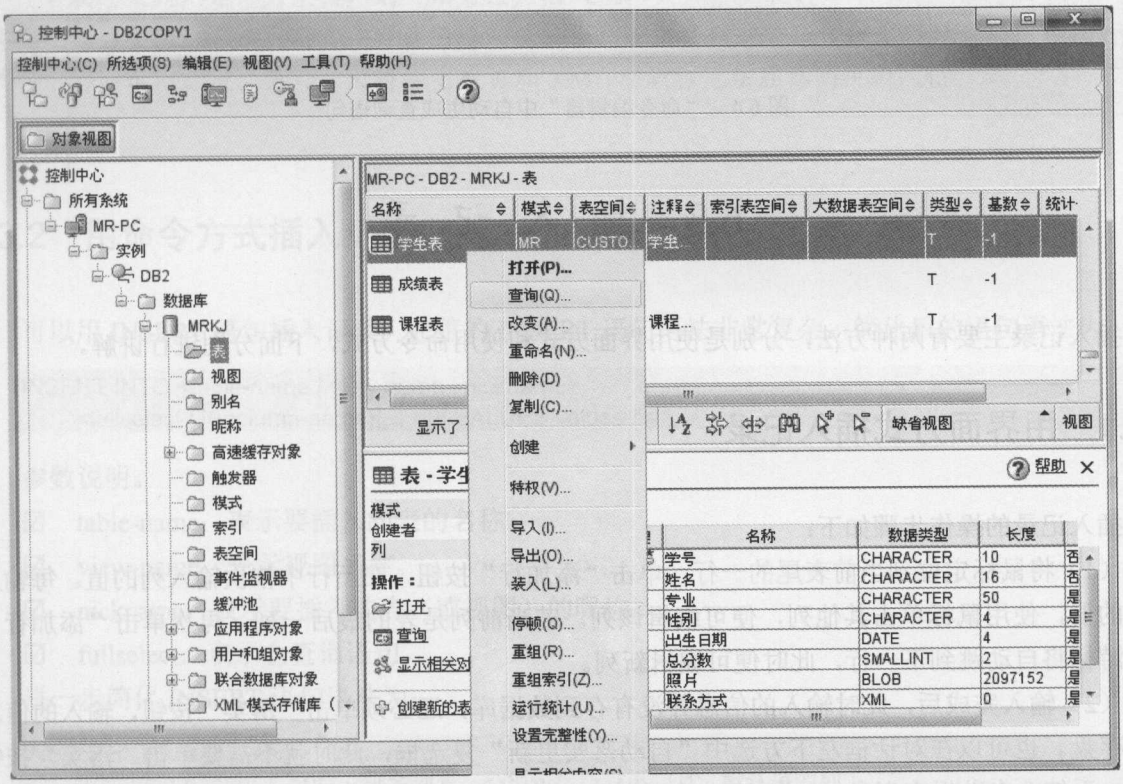


图 6.3　“查询”学生表

（2）此时“控制中心”会增加一个“命令编辑器”选项卡，如图 6.4 所示。同时系统将自动生成的查询语句填写在“SQL 语句窗格”，单击图中▶按钮执行命令。

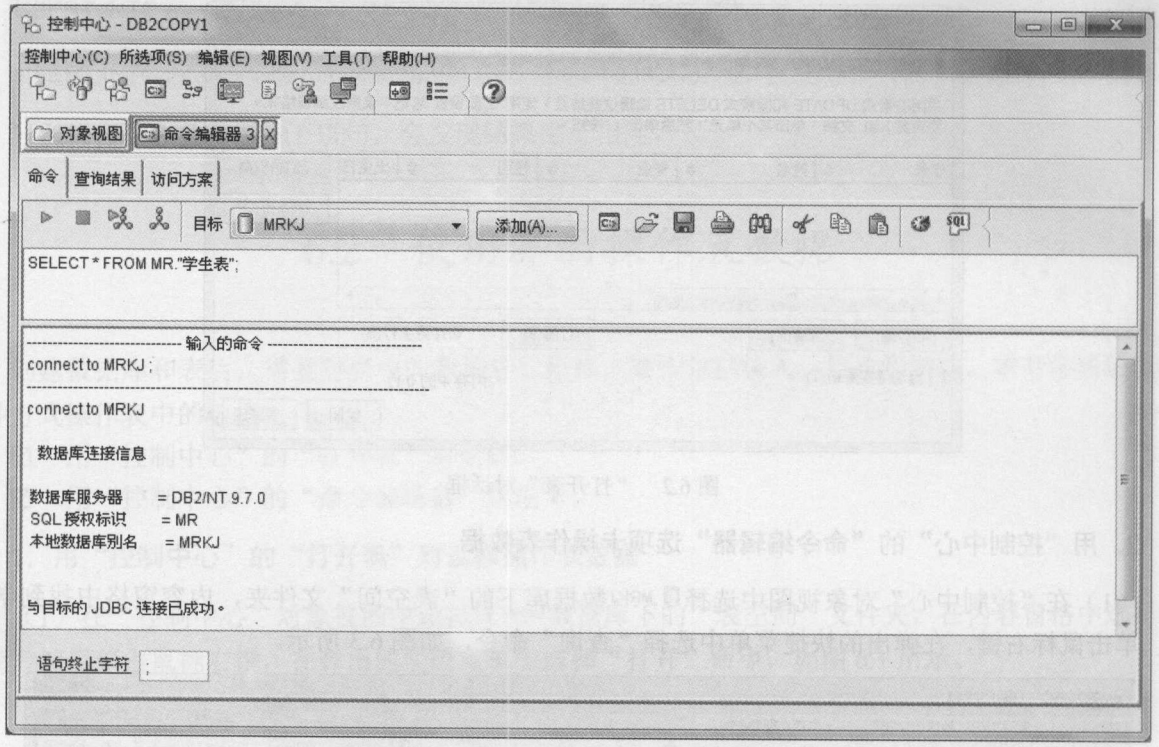


图 6.4 “命令编辑器”中自动生成查询语句

6.3 插入记录

插入记录主要有两种方法，分别是使用界面方式和使用命令方式，下面分别进行讲解。

6.3.1 用界面方式插入记录

插入记录的操作步骤如下：

（1）将鼠标定位到当前表尾的一行，单击“添加行”按钮，在新行中依次输入列的值。每输入完一列的值，使用鼠标单击其他列，便可编辑该列。若当前列是表的最后一列，再次单击“添加行”按钮，游标将自动跳到下一行，此时便可编辑新列。

（2）输入完成后，此时输入的信息并没有存到数据库，还必须单击“落实”按钮，输入的信息才提交存盘。也可以在对话框左下方选中“自动落实更新”复选框，此时就不需要单击“落实”按钮来存盘，系统会自动提交存盘。

下面使用界面方式向“学生表”中添加 3 条记录，插入后的数据如图 6.5 所示。

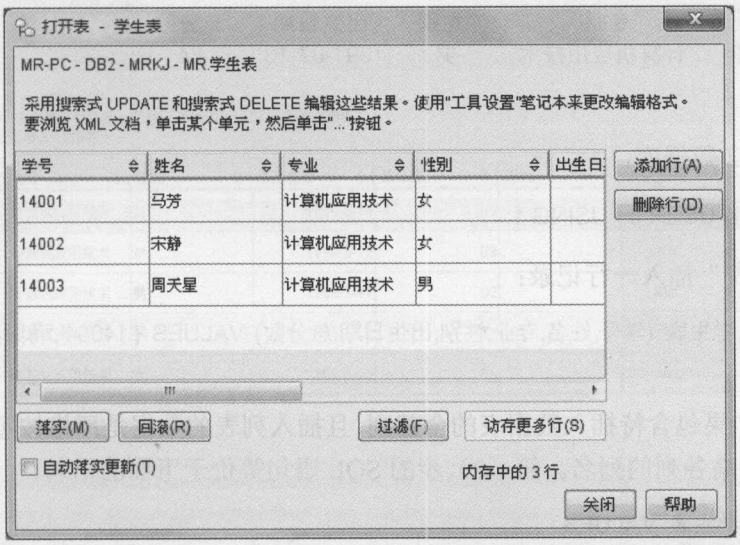


图 6.5　插入记录

注意

若表的某列不允许为空值，则必须为该列输入值，例如学生表的“学号”“姓名”等列；若某些列允许为空值，那么则不需要对该列输入值。如果表中某列有默认值，则不对该列输入值时，在表格中将显示默认值。另外，大对象型数据和 XML 型数据无法在界面中输入值。

6.3.2　用命令方式插入记录

可以用 INSERT 语句插入记录。完整的 INSERT 语句语法非常复杂。简化后的语句语法为：

```
INSERT INTO <table-name | view-name | nickname>
    <fullselect | [({column-name}[,...n])] VALUES values-list>
```

参数说明。

- ☑ table-name：表示要插入的表的名称。
- ☑ view-name：表示视图名称。
- ☑ nickname：表示要插入的表（或视图）的昵称。
- ☑ fullselect：表示全查询语句。

进一步简化 INSERT 语句语法为：

```
INSERT INTO <table-name | view-name>
    [({column-name}[,...n])]  --列名列表，必须包括所有不空且无默认值的列
    VALUES （{value}[,...n]） --列值列表，该处的值的顺序和数据类型与上面列名一一对应
```

【例 6.3】 使用 SQL 语句向“学生表”插入样本数据（**实例位置：光盘\TM\sl\6\3**）。

学号	姓名	专业	性别	出生日期	总分数
14004	蒋阳泉	计算机应用技术	男	1991-07-13	84

实现步骤如下。

（1）连接 MRKJ 数据库：

```
CONNECT TO mrkj USER mr USING 1;
```

（2）向“学生表”插入一行记录：

```
INSERT INTO mr.学生表 (学号,姓名,专业,性别,出生日期,总分数) VALUES ('14004','蒋阳泉','计算机应用技术','男',
'1991-07-13',84);
```

在 SQL 语句中如果包含待插入数据表的全部列，且插入列表的数据是严格按照表的先后顺序排列，这样的情况下可以省略各列的列名。第（2）步的 SQL 语句等价于下面的语句：

```
INSERT INTO mr.学生表 VALUES
('14005','董敏','计算机应用技术','女','1992-8-5',88,NULL,NULL,NULL);
```

注意

上面的语句中将允许为空的 3 个列“照片”，“联系方式”和“备注”都插入了 NULL，这 3 个列是不允许省略的。后面的章节会介绍如何操作 XML 和 BLOB 数据类型的列中的数据。这里之所以将“照片”和“联系方式”都插入了 NULL，是为了暂时降低难度，让读者由浅入深地学习。

（3）使用“命令编辑器”查询“学生表”中的数据，步骤如下。

在“控制中心”中单击按钮，从而启动“命令编辑器”，在“命令编辑器”中输入下面的 SQL 命令：

```
SELECT * FROM mr.学生表;     --查询表中所有数据
```

然后单击按钮，运行后结果如图 6.6 所示。

【例 6.4】 创建一个与“学生表”相同的 student 表，并且向 student 表批量插入总成绩大于 80 的所有学生的信息（**实例位置：光盘\TM\sl\6\4**）。

实现步骤如下。

（1）连接到 MRKJ 数据库：

```
CONNECT TO mrkj USER mr USING 1;
```

（2）创建一个与“学生表”结构相同的 student 表：

```
CREATE TABLE mr.student LIKE mr.学生表;
```

（3）向 student 表批量插入数据：

```
INSERT INTO mr.student SELECT * FROM mr.学生表 WHERE 总分数>=80;
```

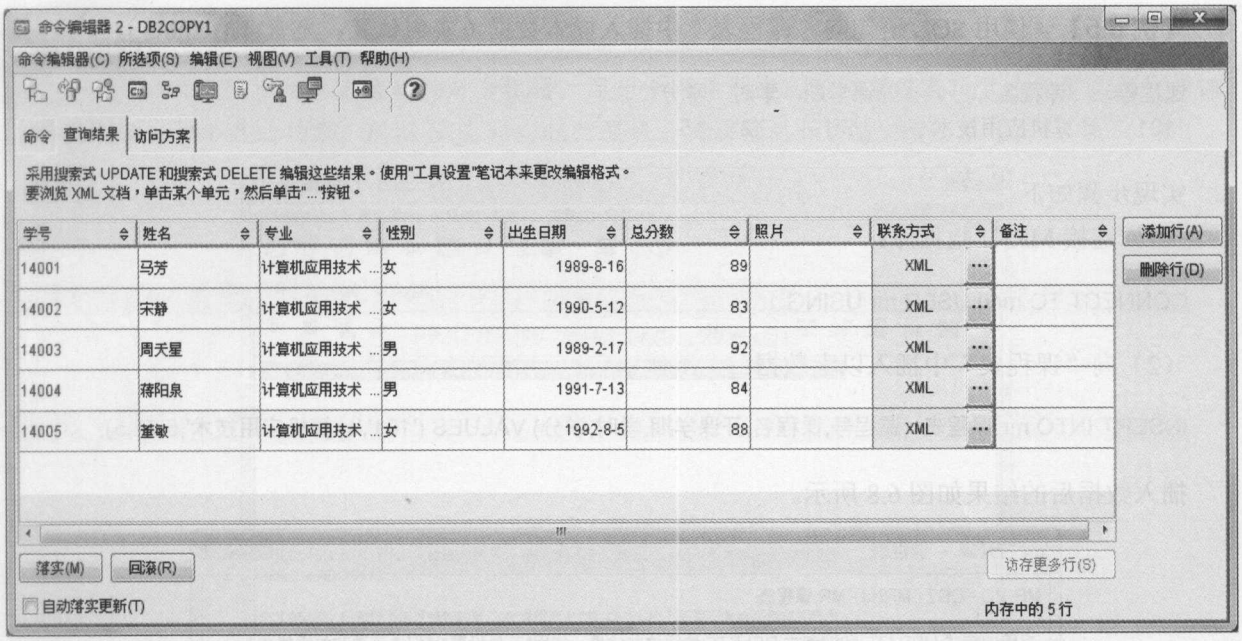


图 6.6　查询表中的数据

（4）在“命令编辑器”中使用下面语句查询 student 表：

```
SELECT * FROM mr.student;
```

查询结果如图 6.7 所示。

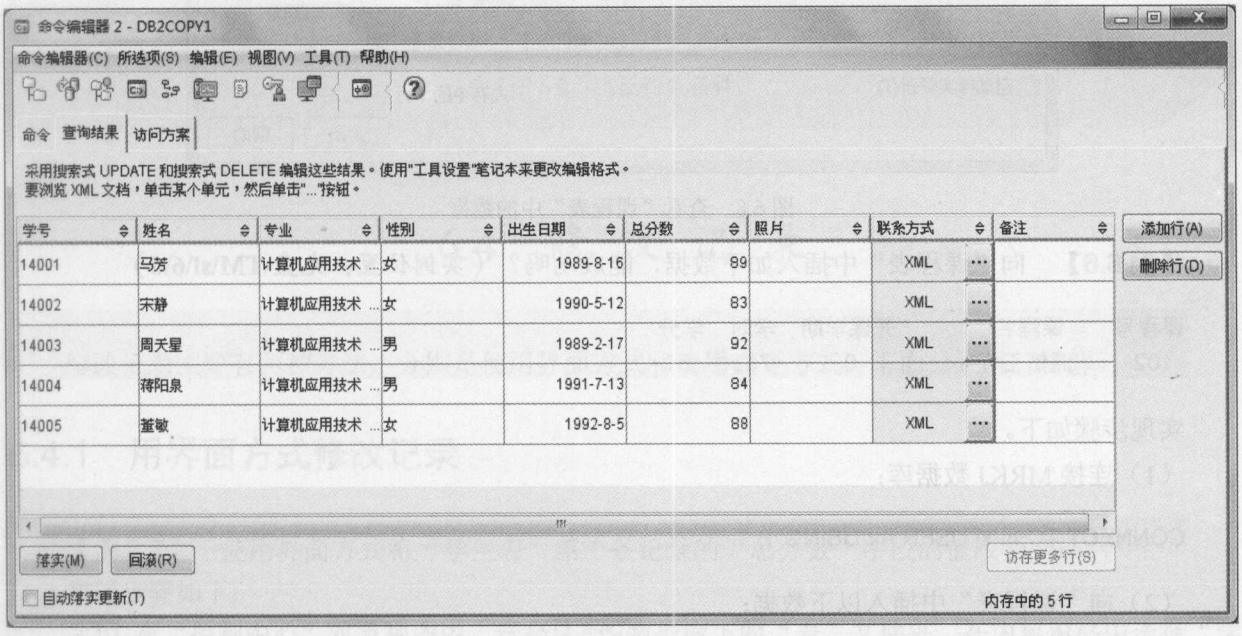


图 6.7　查询 student 表中的数据

【例 6.5】 使用 SQL 语句向“课程表”中插入样本数据（**实例位置：光盘\TM\sl\6\5**）。

课程号	课程名	开课学期	学时	学分
101	计算机应用技术	5	72	5

实现步骤如下。

（1）连接 MRKJ 数据库：

```
CONNECT TO mrkj USER mr USING 1;
```

（2）向“课程表”中插入以上数据：

```
INSERT INTO mr.课程表 (课程号,课程名,开课学期,学时,学分) VALUES ('101','计算机应用技术',5,72,5);
```

插入数据后的结果如图 6.8 所示。

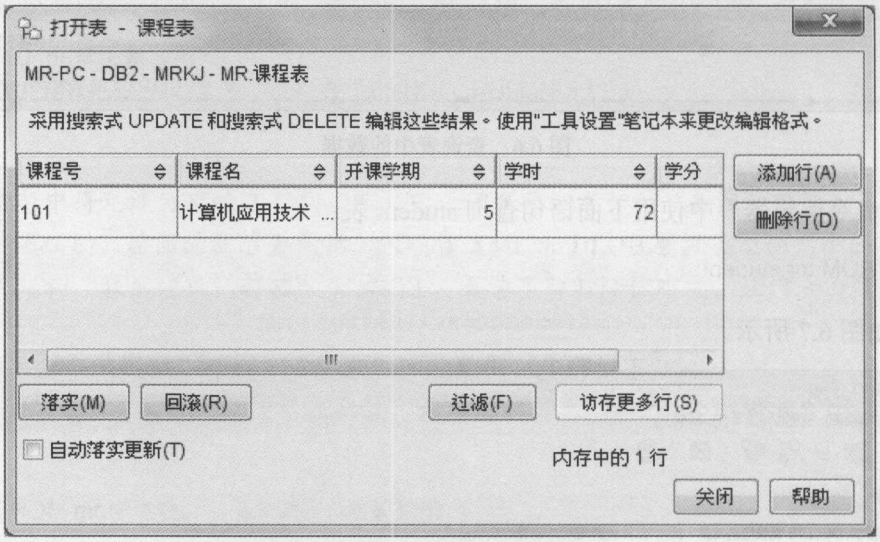


图 6.8 查看“课程表”中的数据

【例 6.6】 向“课程表”中插入如下数据，能成功吗？（**实例位置：光盘\TM\sl\6\6**）

课程号	课程名	开课学期	学时	学分
102	通信工程	0	75	5

实现步骤如下。

（1）连接 MRKJ 数据库：

```
CONNECT TO mrkj USER mr USING 1;
```

（2）向“课程表”中插入以下数据：

```
INSERT INTO mr.课程表 (课程号,课程名,开课学期,学时,学分) VALUES ('102','通信工程',0,75,5);
```

说明

向“课程表”中插入上述数据时会报错。因为开课学期违反了“课程表”上已经定义的“开课学期”必须为 1~8 的约束，所以 SQL 语句执行错误。错误信息如图 6.9 所示。

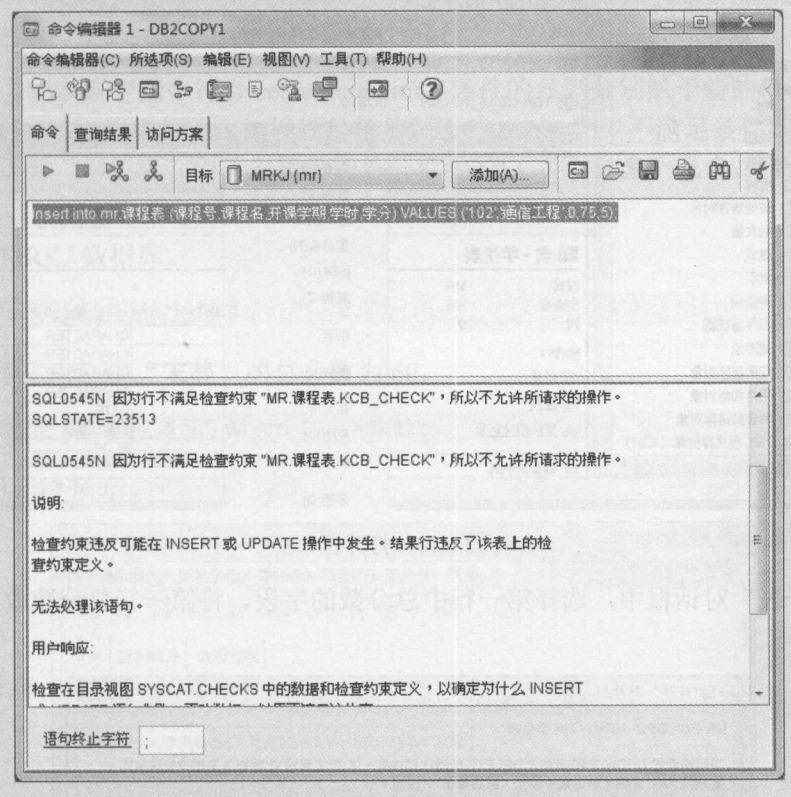


图 6.9　违反约束条件

6.4　修 改 记 录

修改记录主要有两种方法，分别是使用界面方式和使用命令方式，下面分别进行讲解。

6.4.1　用界面方式修改记录

【例 6.7】　使用界面方式将“学生表”第 1 条记录的“总分数”字段的值改为 70。

实现步骤如下：

（1）在“控制中心”对象视图中，选择 MRKJ 数据库下的“表”文件夹，在内容窗格中选择“学生表”，然后单击鼠标右键，在弹出的快捷菜单中选择“打开”命令，如图 6.10 所示。

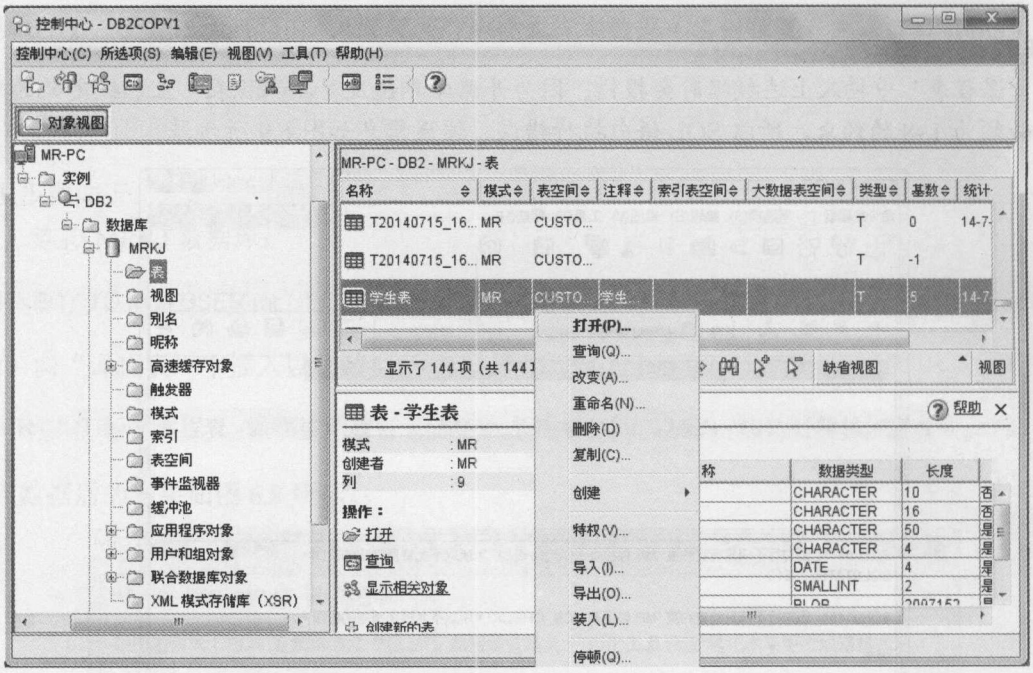


图 6.10　打开表

（2）在“打开表”对话框中，选择第一行中总分数的字段，将第一行总分数改成 70，如图 6.11 所示。

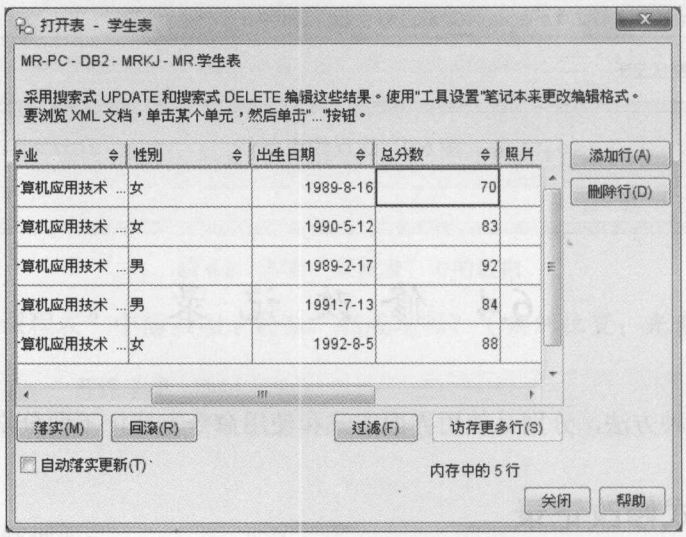


图 6.11　更改后的结果

6.4.2　用命令方式修改记录

可使用 UPDATE 语句修改表中的记录。完整的 UPDATE 语句的语法也比较复杂。简化后的 UPDATE

语句语法如下：

```
UPDATE <table_name | view_name>
  SET {column_name=new_column_value}
    WHERE search_condition;
```

参数说明：

☑ 更新语句会将表或视图中符合 search-condition 条件的各列分别赋予新的值。

☑ search-condition：搜索条件，如果满足条件则返回 true，如果不满足条件则返回 false。

【例 6.8】　更新“学生表”中“宋静”的总分数为 60（**实例位置：光盘\TM\sl\6\7**）。

实现步骤如下：

（1）连接到 MRKJ 数据库。

```
CONNECT TO mrkj USER mr USING 1;
```

（2）更新“学生表”中“宋静”的总分数为 60。

```
UPDATE mr.学生表 set 总成绩=60 where 姓名='宋静';
```

更新结果如图 6.12 所示。

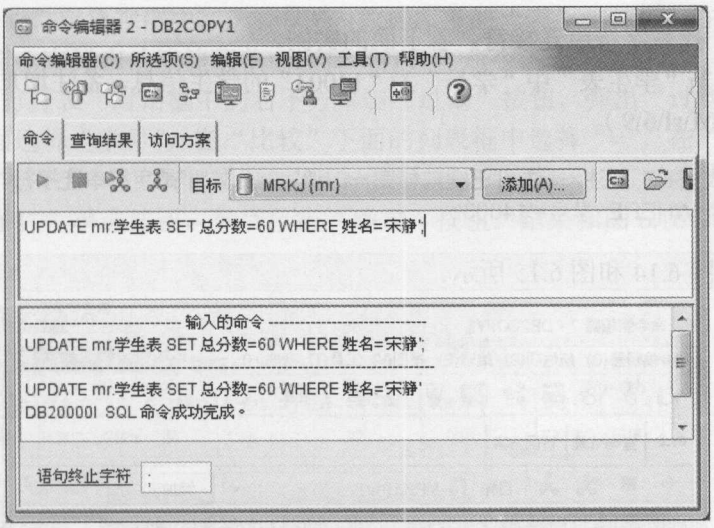


图 6.12　更新语句执行结果

【例 6.9】　更新“学生表”中学号为“14003”的学生信息，总分数增加 4，备注改为“三好学生”（**实例位置：光盘\TM\sl\6\8**）。

实现步骤如下：

（1）连接到 MRKJ 数据库。

```
CONNECT TO mrkj USER mr USING 1;
```

（2）更新“学生表”中学号为“14003”的学生信息，总分数增加 4，备注改为“三好学生”。

```
UPDATE 学生表 SET 总分数=总分数+4,备注='三好学生' WHERE 学号='14003';
```

更新后的结果如图 6.13 所示。

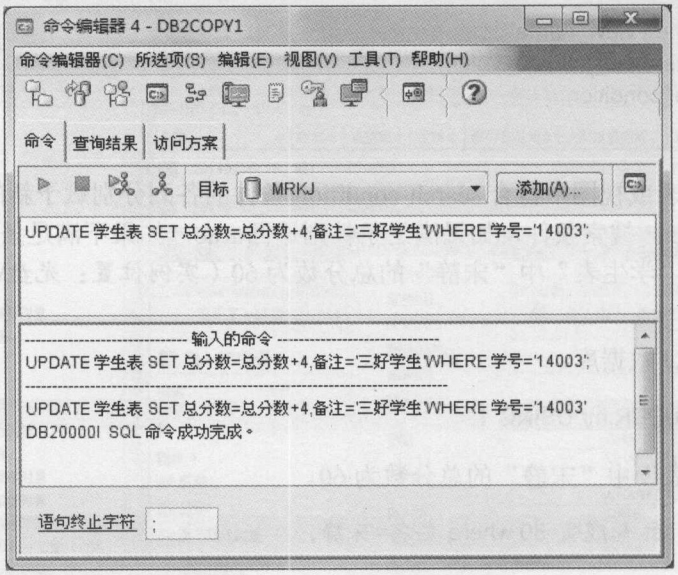


图 6.13　更新语句的执行结果

在数据库中“||”代表字符串连接符。看下面的实例。

【例 6.10】　更新“学生表”中“学号”为“14003”的学生信息，备注增加“已提前修完一门课”（**实例位置：光盘\TM\sl\6\9**）。

```
UPDATE mr.学生表 SET 备注=备注 ||'已提前修完一门课'  --此处"||"是字符串连接符
                 WHERE 学号='14003';
```

更新后的结果如图 6.14 和图 6.15 所示。

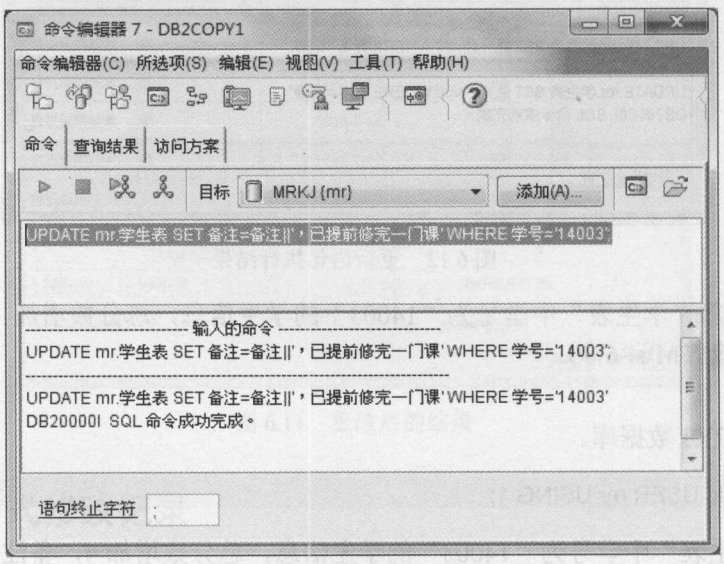


图 6.14　执行更新语句的结果

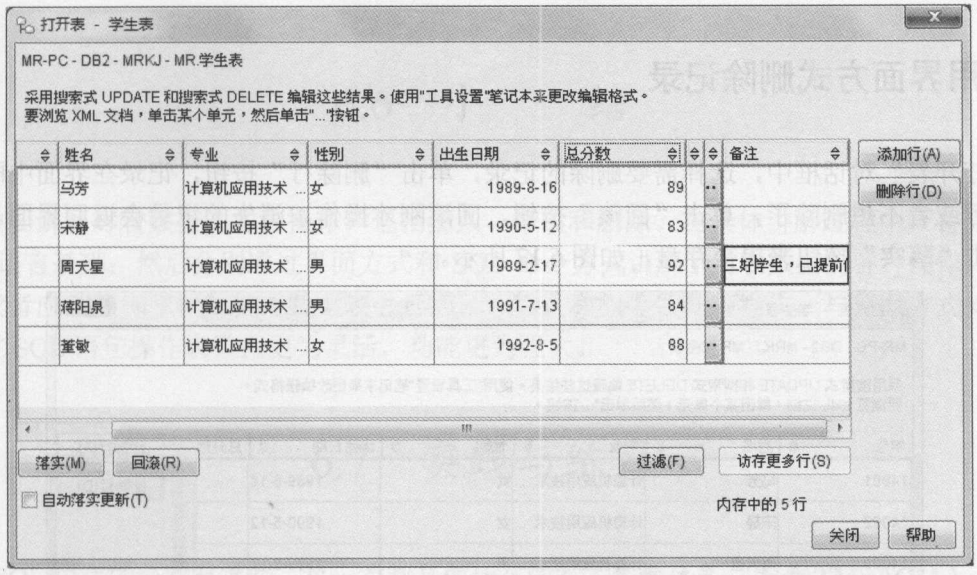


图 6.15　查看结果

从例 6.8 到例 6.10 中的更新操作完成之后，查看更新操作的效果。

（1）在“控制中心”对象视图中，选择 MRKJ 数据库下的“表”文件夹，在内容窗格中找到学生表，然后单击鼠标右键，在弹出的快捷菜单中选择“打开”命令。

（2）在弹出的“打开表”对话框中的右下方单击“过滤”按钮，弹出“过滤”对话框，在该对话框中，在“列”的下面选择“学号”，在“比较”下面的列表框中选择“=”，在“值”的下面的列表框中输入“14003”；在“姓名”一行中的“比较”下方的列表框中选择“=”，在值的下方输入“周天星”，然后选中“满足任何条件”单选按钮，最后单击“确定”按钮。结果如图 6.16 所示。

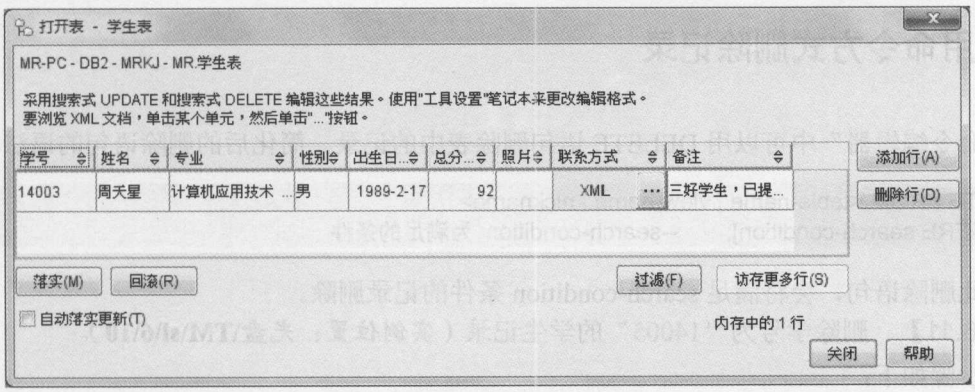


图 6.16　通过过滤条件查询记录

6.5　删 除 记 录

删除记录主要有两种方法，分别是使用界面方式和使用命令方式，下面分别进行讲解。

6.5.1 用界面方式删除记录

在“打开表”对话框中，选择需要删除的记录，单击“删除行”按钮，记录在界面中即可消失。如果删错行或者不想删除了，单击“回滚”按钮，回滚刚才操作，消失的记录会返回界面。如果确认无误，单击“落实”按钮来提交存盘，如图 6.17 所示。

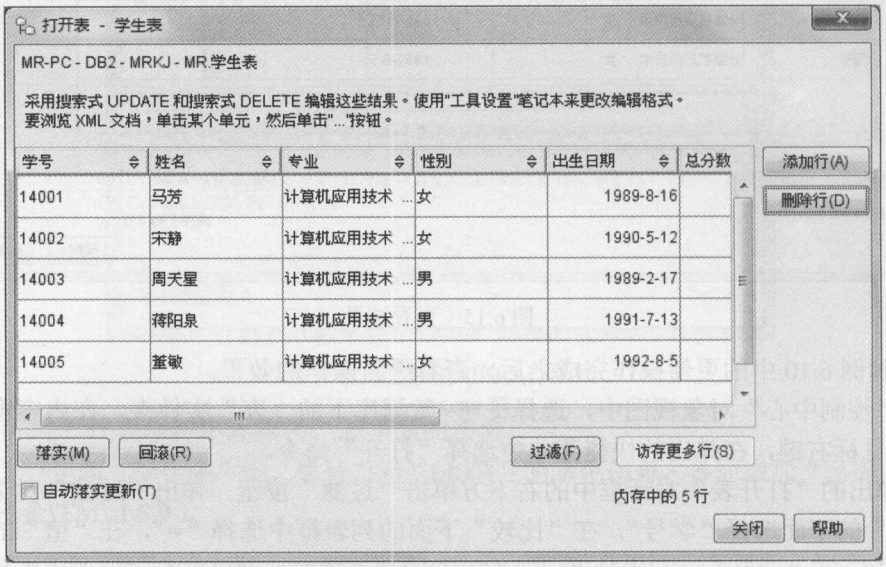


图 6.17 删除数据行

6.5.2 用命令方式删除记录

在“命令编辑器”中可以用 DELETE 语句删除表中的记录。简化后的删除语句的语法格式为：

```
DELETE FROM <table-name | view-name | nickname>
  [WHERE search-condition];      --search-condition 为满足的条件
```

执行此删除语句，会将满足 search-condition 条件的记录删除。

【例 6.11】 删除学号为“14005”的学生记录（**实例位置：光盘\TM\sl\6\10**）。

实现步骤如下：

（1）连接 MRKJ 数据库。

```
CONNECT TO mrkj USER mr USING 1;
```

（2）删除学号为“14005”的学生记录。

```
DELETE FROM mr.学生表
      WHERE  学号='14005';
```

6.6　小　　结

本章主要讲解对表数据的各种操作，包括插入、修改和删除。在具体讲解过程中，首先介绍了基本的 SQL 语言基础；然后分别通过界面方式和 SQL 命令方式讲解了对表数据的各种操作。通过本章的学习，读者应该熟练掌握如何对数据表进行插入、修改和删除数据等操作。与界面方式操作表数据相比，通过 SQL 语句操作表数据更为灵活，功能更为强大。

6.7　实践与练习

（1）SQL 语言的分类有哪些，每一类都是做什么的？（**答案位置：光盘\TM\sl\6\11**）

（2）在界面方式操作与使用 SQL 命令操作数据，两种方法相比较，哪一种功能更强大、更为灵活？举例说明。（**答案位置：光盘\TM\sl\6\12**）

（3）创建一个产品价格表。（**答案位置：光盘\TM\sl\6\13**）

编　号	名　称	价　格
0001	空调	2000
0002	冰箱	4288
0003	彩电	2800

第 7 章

数据库对象

（视频讲解：47 分钟）

前面已经介绍了 DB2 的安装配置和建库、建表以及表的操作。可以说“框架”已经搭建好了，但是对于设计/开发人员来说，更关心的是框架里面该如何“装饰”，即数据库对象如何操作。因为无论是数据库建模、应用开发还是数据库访问，都是围着数据对象展开的。尽管各个数据库之间存在类似的概念，但是对于一些对象的概念和用法还是存在差异的，能够深入理解各个对象的概念和用法是学习 DB2 的基础。

通过阅读本章，您可以：

- 了解数据库模式的概念
- 掌握如何创建模式
- 掌握如何创建表的约束
- 了解视图和昵称
- 掌握序列、自增字段

7.1　数据库对象概述

在数据库创建后，可以根据业务需求来设计和创建数据库对象。可以在 DB2 数据库中创建下列数据库对象：

- ☑ 模式。
- ☑ 表。
- ☑ 索引。
- ☑ 视图。
- ☑ 昵称。
- ☑ 序列。
- ☑ 自增字段。

可以使用图形界面或通过显示执行 SQL 语句来创建这些数据库对象，用于创建这些数据库对象的语句称为“数据定义语言”（DDL），它们通常以关键字 CREATE 或 ALTER 作为前缀。下面的章节会分别对数据库的常用对象进行讲解。

7.2　模　　式

在数据库中每一个表都有自己唯一的名字。但是有时候，一个数据库可能被多个用户使用。例如一个数据库有 A 和 B 两个用户，A 用户有一个表叫 STUDENT，B 用户也想建立一个名字叫 STUDENT 的表，该怎么办呢？

这种情况下就需要创建模式了。模式的英文名是 schema，DB2 中的模式是一个已命名对象的集合。它提供一种方法来按逻辑分组这些对象，这些对象包括表、视图、索引、触发器、函数和包。

模式主要用于：

- ☑ 用来对数据库对象按照功能性质进行逻辑分组。
- ☑ 对数据库的对象按照组进行权限管理。

可以把模式想象成是一个对象的“姓”，而对象的名称则是对象的“名”。例如有两个人，一个人叫作张强，另一个叫作李强，虽然两个人的名字都是“强”，但是他们有不同的姓，这样很容易将他们区分开。数据库对象的命名由模式名加上对象名构成（模式名.对象名）。

因此，如果 A 用户想要创建一个 STUDENT 表，则可以指定这个表模式为 A，也就是 A.STUDENT。而 B 则可以用 B.STUDENT 来命名它的表。而在程序中，如果模式没有被显式指定，那么就会用运行该应用程序的用户名作为默认模式，例如查询：SELECT * FROM STUDENT，当用户 A 执行时就会从 A.STUDENT 中选择数据；而用户 B 执行时就会从 B.STUDENT 中选择。

DB2 数据库中的模式与 Oracle 数据库中的模式有根本差别，Oracle 数据库的模式名是用户名，DB2

中的模式不一定是用户名，因为 DB2 内部没有用户的概念，连接用户必须是操作系统用户。

对于每个数据库，都会创建一组系统模式，并将它们放在 SYSCATSPACE 表空间中，DB2 内部提供了以下几个系统模式用来对相关对象进行分组：

- ☑ SYSIBM 模式下的对象存储的是系统数据字典表，它是基本的系统编目，不建议直接访问它。
- ☑ SYSCAT 模式下的对象是系统视图，可通过这些视图查看各种数据库对象信息。这种模式上的 SELECT 权限被授给 PUBLIC，建议通过它来编目信息。
- ☑ SYSIBMADM 是 V9 引入的系统管理视图模式。
- ☑ SYSSTAT 是统计视图模式，该模式下包含 9 个视图，用来为 DB2 优化器提供统计信息。
- ☑ SYSFUN：用户定义函数。

7.2.1 使用控制中心的方式创建模式

（1）在“控制中心”对象视图中，展开“所有数据库”后选择 MRKJ 数据库，然后展开 MRKJ 数据库便能看到“模式”文件夹，单击“模式”文件夹在内容窗格中会显示所有“模式”，如图 7.1 所示。

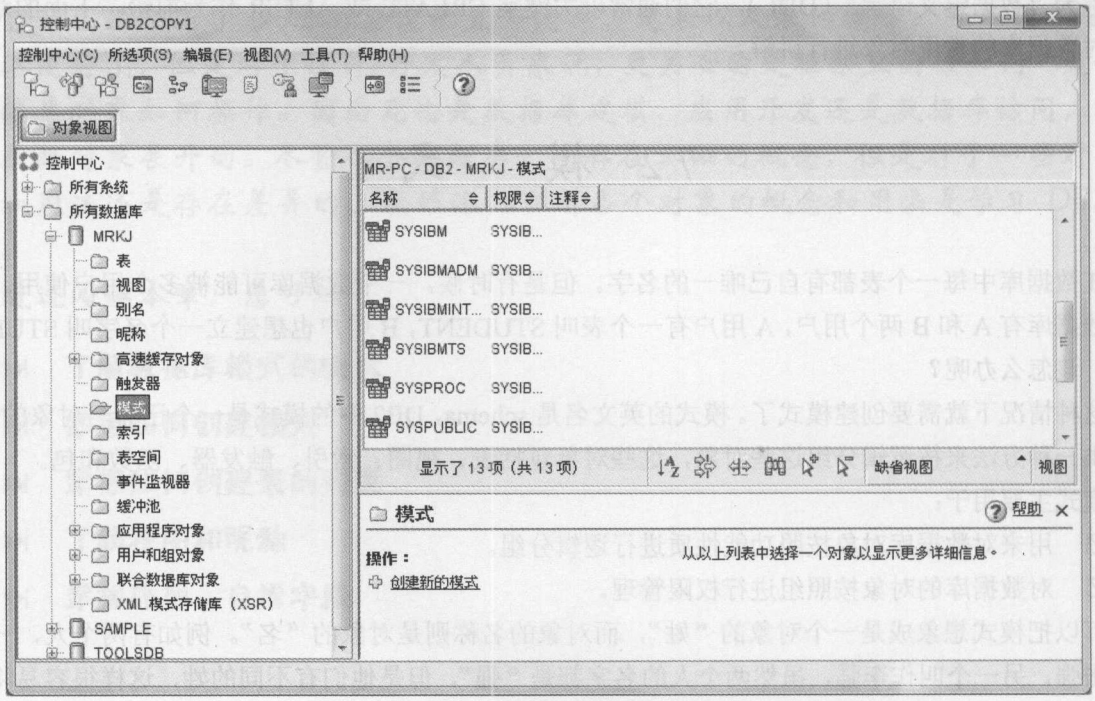


图 7.1 查看数据库中的“模式”

（2）在“模式”文件夹上单击鼠标右键，在弹出的快捷菜单中选择“创建”命令，如图 7.2 所示。此时打开“创建模式”对话框，在该对话框中输入“MRKJ”作为新的模式名称，使用默认的“ADMINISTARTOR”作为权限名，然后单击“确定”按钮（想要看创建“模式”的命令，可以单击“显示 SQL”），如图 7.3 所示。

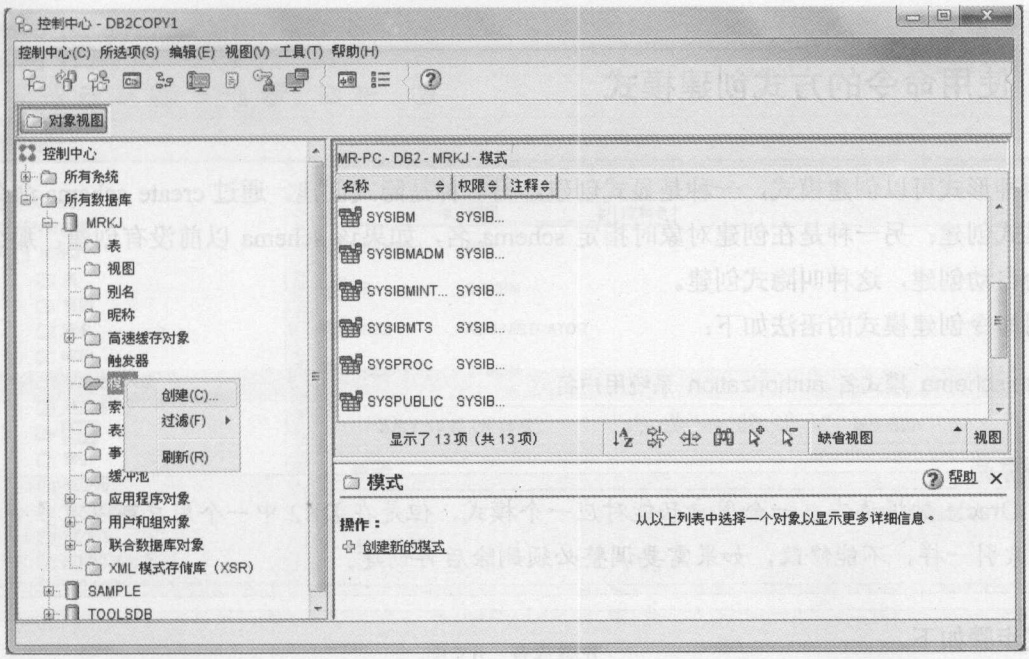


图 7.2　创建“模式”

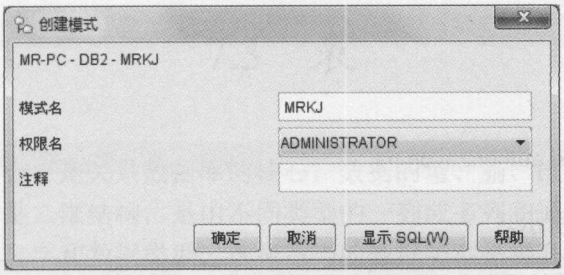


图 7.3　“创建模式”对话框

（3）单击“确定”按钮之后，在“内容窗格”中就可以看到 MRKJ 模式，如图 7.4 所示。

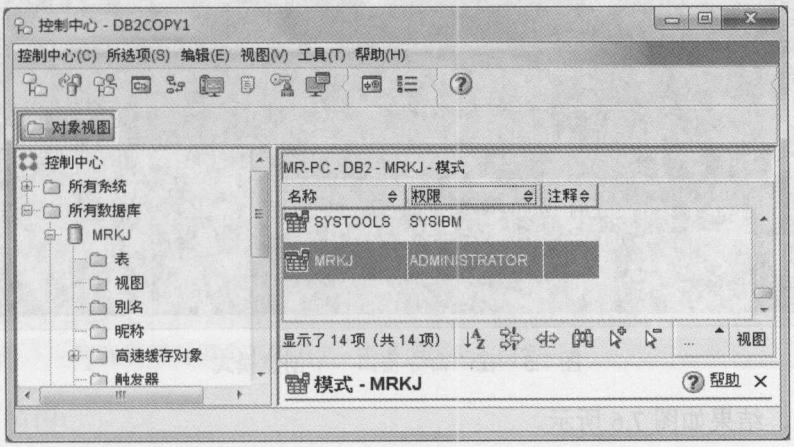


图 7.4　在内容窗格中查看模式

7.2.2 使用命令的方式创建模式

有两种形式可以创建模式，一种是显式创建，另一种是隐式创建。通过 create schema 命令创建，这种叫显式创建；另一种是在创建对象时指定 schema 名，如果该 schema 以前没有创建，那么在创建对象时会自动创建，这种叫隐式创建。

使用命令创建模式的语法如下：

```
create schema 模式名 authorization 系统用户名
```

注意

在 Oracle 数据库中，一个用户只能对应一个模式，但是在 DB2 中一个用户可以有多个模式。模式和索引一样，不能修改，如果需要调整必须删除后再创建。

实现步骤如下。

（1）打开“命令窗口”连接 MRKJ 数据库：

```
db2 connect to mrkj
```

（2）创建模式：

```
db2 create schema db2mrkj authorization mr
```

整个输入过程如图 7.5 所示。

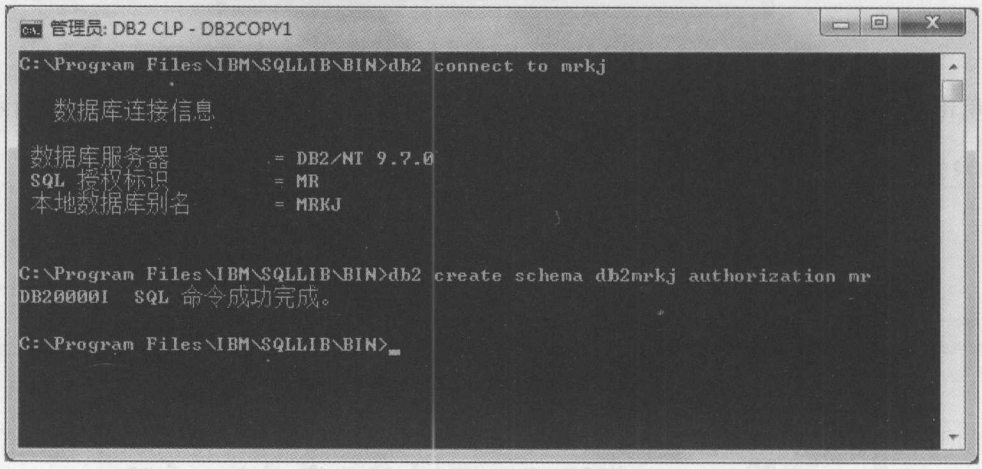


图 7.5 在“命令窗口”中创建模式

创建完模式后，结果如图 7.6 所示。

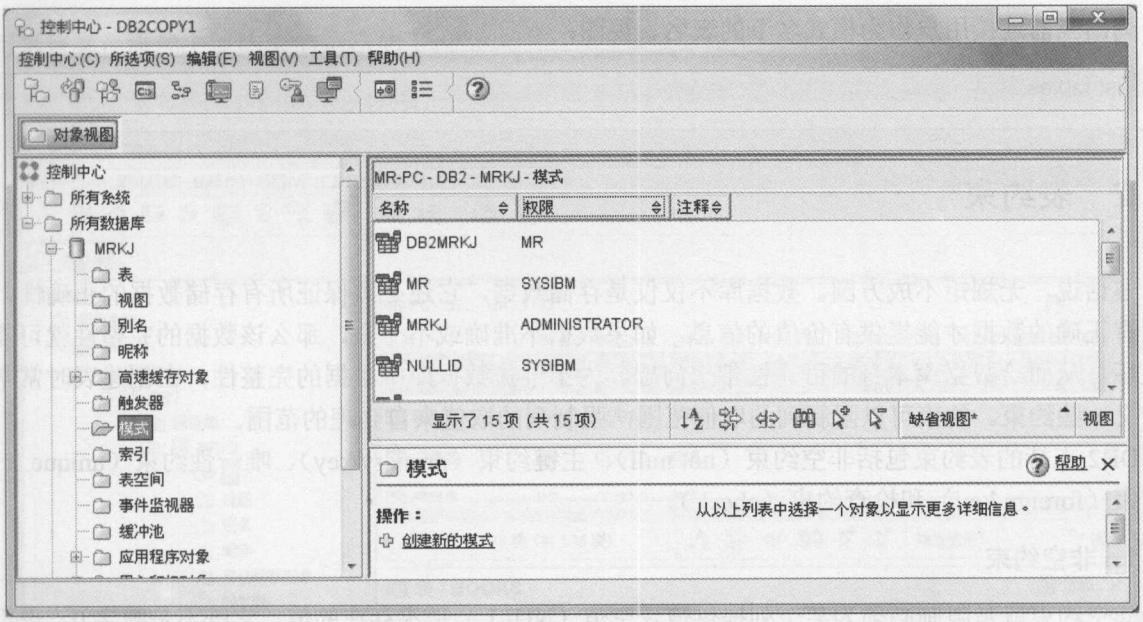


图 7.6　查看模式

7.3　表

表是数据记录的无序集合，是关系数据库的核心，数据的增、删、改、查、应用的数据模型设计都是围绕着表来进行的。表是二维结构，是由不同类型的一列或多列和通常被认为是记录的行组成。列也叫字段，每个字段都有名字和数据类型，选择适当的数据类型是提升性能的前提。所有数据都存储在数据库的表中。一个数据库有一个表集，称为系统目录表（System Catalog Table），它保存关于数据库中所有对象的信息。数据库中定义的每个表在目录表 SYSCAT.TABLES 中都有相应的一行。数据库中每个表的每一列在 SYSCAT.COLUMNS 中都有相应的一行。

7.3.1　表的维护

列出当前用户的表：

```
list tables for schema <schema_name>      --可查看某个模式下的表名、视图和 alias 别名
```

列出数据库中定义的所有表：

```
list tables for all      --可查看所有模式下的表名、视图和 alias 别名
```

列出指定表的结构：

```
describe table tablename      --可查看指定表的结构
```

列出当前连接用户作为模式名下的表名、视图：

```
list tables
```

7.3.2 表约束

俗话说，无规矩不成方圆。数据库不仅仅是存储数据，它还必须保证所有存储数据的正确性，因为只有正确的数据才能提供有价值的信息。如果数据不准确或不一致，那么该数据的完整性就可能受到破坏，从而给数据库本身的可靠性带来问题。为了维护数据库中数据的完整性，在创建表时常常需要定义一些约束。约束可以限制列的取值范围，强制列的取值来自合理的范围。

DB2 支持的表约束包括非空约束（not null）、主键约束（primary key）、唯一性约束（unique）、外键约束（foreign key）和检查约束（check）。

1. 非空约束

非空约束就是限制必须为某个列提供值。空值（NULL）是不存在的值，它既不是数字 0，也不是空字符串，而是不存在、未知的情况。

在表中，若某些字段的值是不可缺少的，就可以为该列定义非空约束。这样当插入数据时，如果没有为该列提供数据，系统就会出现一个错误消息。

如果某些列的值是可有可无的，那么可以定义这些列允许空值。这样，在插入数据时，就可以不向该列提供具体的数据（在默认情况下，表中的列是允许为 NULL 的）。如果某个列值不允许为 NULL，就可以使用 NOT NULL 来标记该列，下面来看几个相关例子。

【例 7.1】 在“命令编辑器”中，创建 Books 表，要求 BookNO（图书编号）、ISBN 和 PublisherNO（出版社编号）不能为空值，代码如下，运行结果如图 7.7 所示（**实例位置：光盘\TM\sl\7\1**）。

```
create table Books(
    BookNo character not null,          --图书编号，不为空
    BookName character,                 --图书名称
    Author character,                   --作者
    SalePrice decfloat,                 --定价
    PublisherNo character not null,     --出版社编号
    PublisherDate date,                 --出版日期
    ISBN character not null             --ISBN，不为空
)IN customdataspace1;
```

2. 主键约束

主键约束用于唯一地标识表中的每一行记录。在一个表中，最多只能有一个主键约束，主键约束既可以由一个列组成，也可以由两个或两个以上的列组成（这种称为组合键）。对于表中的每一行数据，主键约束列都是不同的，主键约束同时也具有非空约束的特性。

如果主键约束由一列组成，该主键约束被称为行级约束。如果主键约束由两个或两个以上的列组成时，则该主键约束被称为表级约束。若要设置某个或某些列的主键约束，通常使用 PRIMARY KEY

语句来定义，添加主键约束的语法如下：

```
alter table  表名  add primary key (列名);
```

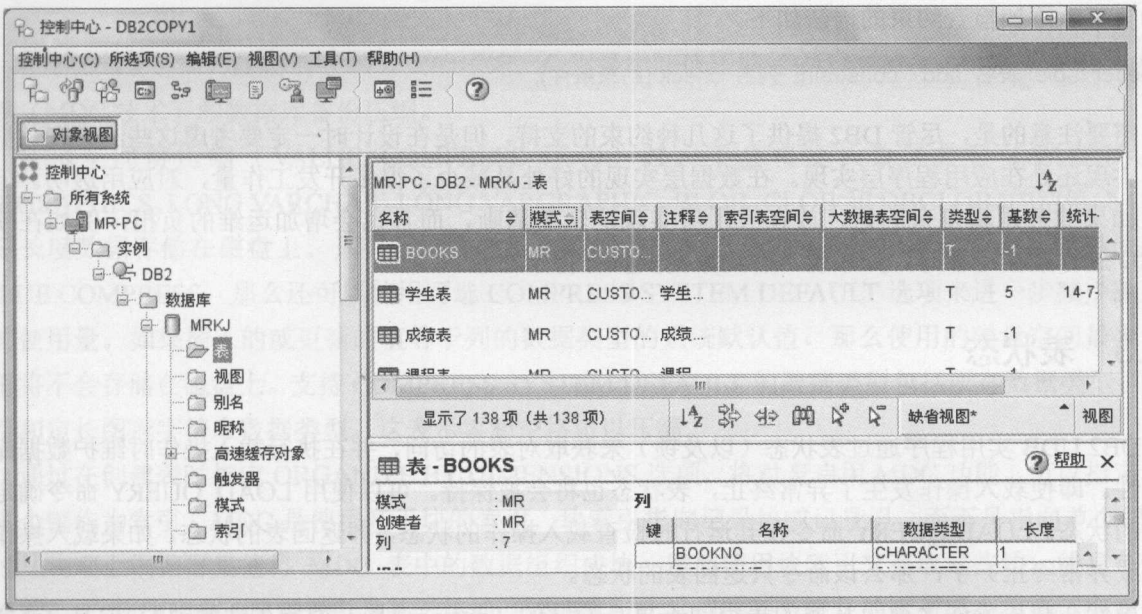


图 7.7　带有非空约束的 Books 表

3．唯一性约束

唯一性（UNIQUE）约束强调所在的列不允许有相同的值。但是它的定义要比主键约束弱，即它所在的列允许有空值（主键约束列是不允许为空值的）。唯一性约束的主要作用是在保证除主键列外，其他列值的唯一性。

在一个表中，根据实际情况可能有多个列的数据都不允许存在相同值。例如，各种“会员表”的 QQ、Email 等列的值是不允许重复的（但是用户可能不提供，这样就必须允许为空值），但是由于在一个表中最多只能有一个主键约束存在，那么如何解决这种多个列都不允许重复数据存在的问题呢？这就是唯一性约束的作用。若要设置某个列为 UNIQUE 约束，通常使用 UNIQUE 标记该列，添加唯一性约束的语法如下：

```
alter table  表名  add unique (列名);
```

4．外键约束

外键约束比较复杂，一般的外键约束会使用两个表进行关联（当然也存在同一个表自连接的情况）。外键是指“当前表”（即外键表）引用“另外一个表”（即被引用的表）的某个列或某几个列，而“另外一个表”中被引用的列必须具有主键约束或者唯一性约束。在“另外一个表”中，被引用列中不存在的数据不能出现在“当前表”对应的列中。一般情况下，当删除被引用表中的数据时，该数据也不能出现在外键表的外键列中。如果外键列存储了被引用表中将要被删除的数据，那么对被引用的删除操作将失败，添加外键约束的语法如下：

```
alter table  表名  add foreign key (列名) references  表名  on delete cascade;
```

5. 检查约束

检查约束：指定某个字段内容必须在约束范围内，如定义的年龄字段不能为 200，银行卡的余额不能为负值。添加检查约束的语法如下。

```
alter table 表名 add  constraint 列名 check (约束条件);
```

需要注意的是，尽管 DB2 提供了这几种约束的支持，但是在设计时一定要考虑这些约束在数据库层面实现还是在应用程序层实现。在数据层实现的好处是减少了很多开发工作量，对应用透明。但缺点是如果业务逻辑复杂的话，放在应用层实现逻辑更清晰，而且也会增加运维的负担，例如在使用 LOAD 进行数据迁移时，如果表上有约束需要执行特别检查。

7.3.3 表状态

DB2 UDB 实用程序通过表状态（以及锁）来获取对表的访问，并在执行载入操作时维护数据库的一致性。即使载入操作发生了异常终止，表状态也将会被保持。可以使用 LOAD QUERY 命令确定特定表的状态。LOAD QUERY 命令在其运行时检查载入操作的状态，并返回表的状态。如果载入操作完成（或异常终止）了，那么该命令只返回表的状态。

虽然在载入操作之前，从属的表空间不再是静默的（quiesce 是一种持久性的锁），但是 Load in Progress 表空间状态会在执行载入操作时组织对从属表进行备份。

多种表状态如下。

- ☑ Check Pending：如果在一个表上定义了表检查约束，但还未验证新数据与那些已定义约束的兼容性，那么该表就处于这种状态。例如，当 DB2 load 实用程序开始在定义了表检查约束的表上执行载入 操作时，就将表的状态设置为 Check Pending 状态。如果想使该表恢复为 Normal 状态，则需要执行一条 SET INTEGRITY 语句。
- ☑ Load Pending：如果在可提交数据之前，表上的正在执行的载入操作被异常终止，那么该表就处于这种状态。若要使该表恢复 Normal 状态，则需要调用 load terminate、load restart 或 load replace 操作。
- ☑ Load in Progress：这是一种只在执行载入操作期间才有效的临时状态。
- ☑ Normal：如果表不是处于任何其他（异常的）表状态，那么它就处于 Normal 状态。Normal 状态是表被创建之后的初始状态。
- ☑ Not Load Restartable：当执行完前滚操作，接着出现一个失败的载入操作，而该操作未被成功地重新启动或者终止时，表就处于这种状态。该表还将处于 Load Pending 状态。若要使该表恢复 Normal 状态，则需要执行一条 LOAD TERMINATE 命令。
- ☑ Read Access Only：在执行载入操作时，如果指定了 ALLOW READ ACCESS 选项，那么表就处于这种状态。Read Access Only 是一个临时状态，它允许其他应用程序和实用程序访问在执行载入操作之前就存在的数据。
- ☑ Unavailabel：当前滚一个无法恢复的载入操作时，表就处于这种状态；这样的表只能被删除，或者从备份映像恢复它。

7.3.4　表压缩

在 DB2 V9 之前，DB2 有 3 种方式的压缩，分别是空间值压缩（VALUE COMPRESS 子句）、索引压缩（MDC 技术）和数据库备份压缩。

使用空间值压缩（VALUE COMPRESS 子句）时，不会将变长数据类型（VARCHAR、VARGRAPHICS、LONG VARCHAR、LONG VARGRAPHIC、BLOB、CLOB 和 DBCLOB）的空值（NULL）和零长度数据存储在磁盘上。只有与这些数据类型相关联的开销值才会占用磁盘空间。如果使用了 VALUE COMPRESS，那么还可以使用可选 COMPRESS SYSTEM DEFAULT 选项来进一步减少磁盘空间的使用量。如果插入的或更新的值等于列的数据类型的系统默认值，那么使用的磁盘空间最少。默认值将不会存储在磁盘上。支持 COMPRESS SYSTEM DEFAULT 的数据类型包括所有数据类型、定长字符和定长图形字符串数据类型。这表示零和空格可以压缩。

通过在创建表时指定 ORGANIZE BY DIMENSIONS 选项，将对表启用 MDC 功能，可以指定一个或多个键作为索引。MDC 是使用基于块的索引，块索引指向记录块或记录组，而不是指向单个记录。通过从物理上根据群集值将 MDC 表中的数据组织成块，然后使用块索引来存取这些块，能显著地改善性能。在 MDC 中，群集索引是基于块的。这些索引比常规的基于记录的索引要小很多，因此，占用的磁盘空间更少，并且扫描时速度会更快。

如果数据库中的数据量比较大，可以在备份数据库时通过指定 COMPRESS 选项使用备份压缩功能，从而使备份映像文件需要的存储空间最小。

DB2 V9 新增了可用来压缩数据对象的基于字典的行压缩功能。在压缩数据时，通过使用较少的数据库页来表示相同数据，从而达到节省磁盘存储空间的目的。对于那些行中包含重复模式的大型表，将能从此功能中受益。数据行压缩（COMPRESS 子句）可与现有的空间值压缩（VALUE COMPRESS 子句）一起使用。对于使用行压缩的表，查询性能可能有所提高。可能需要更少的 I/O 操作来访问压缩数据，并且在压缩后，可以将更多数据高速缓存在缓冲池中。由于用户数据压缩在日志记录内，因此日志记录可能会变小。对于 UPDATE 日志记录，则可能不会出现这种情况。与行压缩关联的成本取决于压缩和解压缩数据所需的额外 CPU 周期。在访问行中的数据时，压缩和解压缩是以行为单位执行的。要评估使用行压缩后存储器的节省情况，可使用 DB2 INSPECT 联机实用程序的 ROWCOMPESTIMATE 选项。在启用了表的 COMPRESS 属性并创建了压缩字典之后，才可压缩行。可通过 CREATE 或 ALTER TABLE 语句来设置 COMPRESS 属性。可使用 REORG TABLE 命令来创建压缩字典。在处理 REORG 命令时，现有的所有表行都要被压缩。数据行压缩不适用于索引、LOB、LF 或 XML 对象。

在将表存储在磁盘上时，如果对数据行、空值和系统默认值使用诸如压缩之类的功能，则表可能占用较少的空间。通过数据压缩，可以使用较少的数据库页来存储数据，从而节省磁盘存储空间。由于每页可以存储更多的逻辑数据，因此访问同样多的逻辑数据时需要读取的页数将会少一些。这意味着压缩还可以节省磁盘 I/O。I/O 速度也会加快，因为可以将更多的逻辑数据高速缓存在缓冲池中。

表压缩虽然减少存储空间，也节省了 I/O 和内存占用，但压缩和解压需要一定的 CPU 资源，因此

在 CPU 利用率较高的系统中不建议使用。

7.3.5 表分区

表分区（Table Partition），也叫范围分区，是 DB2 V9 版本引入的功能强大的特性。在 DB2 V9 之前，对一些大表，出于对性能和数据容量限制的考虑，我们通常会把大表分拆成一些小表，再用 UNION ALL 视图的方式联合起来。DB2 V9 在数据容量方面有了重大突破，MS 表空间的新默认类型是“大型”，当使用“大型”MS 表空间时，单表的最大容量限制是 16384GB，而不再是以前的 512GB。DB2 V9 新增了表分区功能，使得用户对大表的性能和数据容量的顾虑都不再存在。使用表分区，能够创建非常大的表，通过跨多个存储器对象划分表数据，可显著增大表，一个表最多可以有 32K 个数据分区。使用表分区分隔数据能够避免扫描不相关的数据，从而提高查询处理性能。DB2 优化器从查询操作中去除不相关的分区。

表分区功能是一种数据组织方案，根据一个或多个表列中的值将表数据划分到多个称为数据分区或范围的存储对象中。每个数据分区都是单独存储的。这些存储器对象可位于不同的表空间和/或相同的表空间中。跨多个存储器对象对表数据进行分区的能力为数据库提供了更高的可伸缩性和灵活性，同时提高了性能和控制能力。表分区可大幅度减少管理庞大数据库所需的维护工作，并可有效增加单个表的潜在大小。表和索引自动重组的新策略选项使用户能够更有效地管理 DB2 服务器对表和索引的自动重组。表分区使用户能够定义表数据的范围，以便单独保存每个范围，用户可基于表中的日期列，按月对表进行分区。每个范围（称为数据分区）与单个存储器对象对应。这些存储器对象可位于不同的表空间和/或相同的表空间中。由于可以对单个数据分区执行管理任务，将很耗时的维护操作分成一系列较小的操作来执行，从而使管理工作更为灵活。例如，可备份和复原单个数据分区而不是整个表。DB2 V9 新增的表分区功能增强了对索引位置的细化控制，可将索引置于不同表空间并单独管理它们。使用 ALTER TABLE 语句的 ATTACH PARTITION 和 DETACH PARTITION 子句可进行快速及方便的数据转入或转出。此功能在数据仓库环境中特别有用，在此环境中经常需要装入或删除数据以运行决策支持查询。能够将表分区与其他数据组织方案组合在一起。通过将表分区与数据分区功能（DPF）一起使用，可跨数据库分区均匀地分布数据范围，以利用 DPF 的查询内并行性和数据库分区负载均衡功能。将表分区与多维集群（MDC）一起使用时，能够对同一表扩展数据块中在多个维上具有类似值的行进行分组。

表分区的优点如下：

- ☑ 表数据的卷入卷出。
- ☑ 更好的查询性能。
- ☑ 支持大表。
- ☑ 更有效的使用层次化管理。
- ☑ 容易管理。

举例来说，一张表存储了 20 年的数据，若想要统计最近几个月的数据，如果采用传统的表结构存

储，那么仍然要访问 20 年的数据才能够获得结果。如果按照月份对表进行分区，那么访问时，DB2 优化器会过滤其他月份，而只计算这几个月的分区数据。因此会带来性能上的巨大提升。另外，采用分区表使数据的归档管理变得简单。

7.4　索　　引

对于大多数用户来说，当问起他们 DB2 中提升应用程序性能最有效的武器是什么，得到的回答很多都是“加索引”。这个答案理论上没有错误，很多时候合适的索引可以成千上万倍地提高系统的性能。但是，为什么加上了索引，系统的性能就奇迹般地提高了？怎么知道一个索引是不是最合适？索引是不是越多越好？下面将为您揭开迷雾。

7.4.1　B+树结构

从数据结构上，DB2 中的索引采用的是 B+树结构。相信学过《数据结构与算法》的读者对 B+树一定不会陌生，如果没学过也不要紧，下面为读者简单介绍一下这种数据结构。

什么是 B+树呢？B+树有什么好处呢？我们先从二叉搜索树谈起。抛开数据库不谈，当想要提高一个算法效率时，很大的一个方面就是尽量在访问判定数据之前就将部分数据排除在外，减少搜索范围，二叉搜索树就是这样一种算法，树中的每一个节点，其左子树一定比右子树小，如图 7.8 所示。

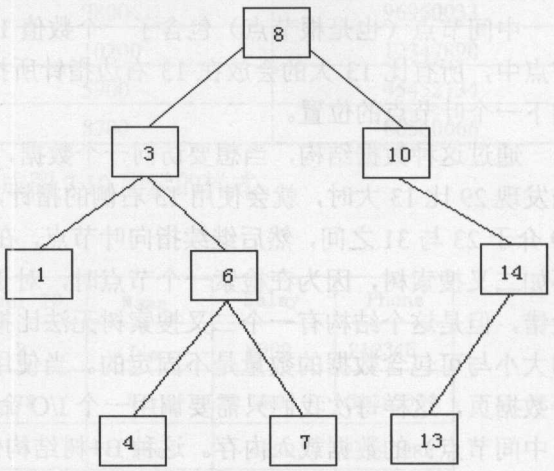


图 7.8　二叉搜索树

在图 7.8 中，如果我们要搜索数字 4。那么当进入树时，首先判断 4 与 8 的关系，如果 4 比 8 小，那么继续搜索左子树。当发现左子树为 3，而 4 比 3 大，则搜索 3 的右子树，直到找到该节点，或者遇到叶节点。

通过这种方式，可以“越过”大部分节点，只搜索树中有限的几个节点，就可以判断出一个数值是否在这个集合之内。

二叉搜索树在数据检索方面与普通的线性检索相比有着无可比拟的优势。但是，对于数据库系统来说，二叉搜索树又有着自己的不足。其中最重要的一点就是：二叉搜索树假设所有的数据都存在内存中，也就是说，访问每一个节点的效率都是等价的。但是在现实生活当中，操作系统的物理内存是有限的，用户的数据量可能是近乎无限大的，那么不可能将所有需要检索的数据都放入内存。因此，在磁盘上的一部分数据，其访问效率将会慢于内存检索几万倍。这样，我们必须有一种方式能够尽量减少磁盘的访问，而不是像二叉搜索树一样可能使用到任何一个节点。于是 B+树就诞生了。实际上

B+树是 B 树的一个变形，我们在这里不讨论 B 树，只关注 B+树。

相比起二叉搜索树，B+树的每一个节点存放着超过一组数据，如图 7.9 所示。所有的最终数据只存放于叶节点，中间的节点存放的是该其子节点的上限或者下限。

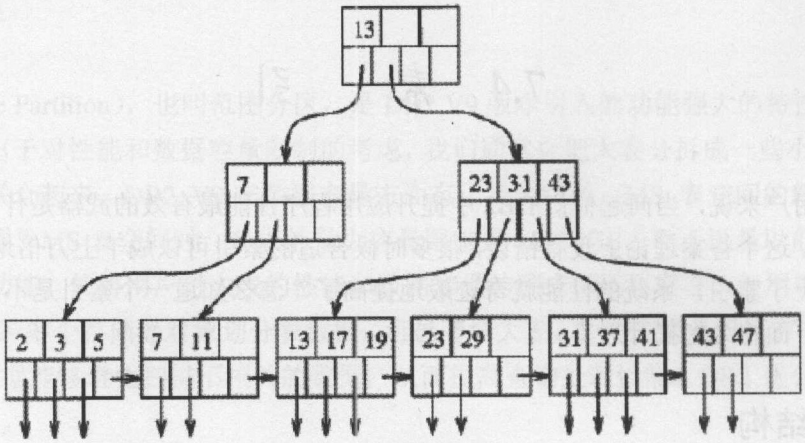


图 7.9　B+树结构

在图 7.9 中，有从 2 到 47 之间的几个数据。在这个 B+树中，每一个节点存放 3 组数据，而根节点下存放两个指针，分别指向两个存放着真实数据的叶节点。

中间节点（也是根节点）包含了一个数值 13。所有比 13 小的数据会放在 13 左边指针所指向的叶节点中，所有比 13 大的会放在 13 右边指针所指向的叶节点中。每一个叶节点末端都有一个指针，指向下一个叶节点的位置。

通过这种数据结构，当想要访问一个数据，例如 29，首先判断这个数值与根节点中数据的关系，当发现 29 比 13 大时，就会使用 13 右侧的指针，得到叶节点，然后在这个叶节点中判断 29 是否存在，29 介于 23 与 31 之间，然后继续指向叶节点，在叶节点中判断 29 是否存在。读者可能会说，这个方法不如二叉搜索树，因为在检索一个节点时，对于其中的多个数据，需要使用线性搜索方式。这样理解没错，但是这个结构有一个二叉搜索树无法比拟的优势，就是每一个节点可以包含多个数据，而节点的大小与可包含数据的数量是不固定的。当使用磁盘时，我们可以定义每一个节点的大小为一个或多个数据页，这样每次我们只需要调用一个 I/O 命令，就可以将若干个相近（对于叶节点），或者可搜索（中间节点）的数据载入内存。这种 B+树结构便是索引的雏形。

7.4.2　索引的好处

读者也许还不理解，为什么 B+树能够快速访问数据呢？答案便在于叶节点中每一个数据所包含的信息。

一般说来，搜索树除了判定某些数值是否在数据集合之内外，更重要的一个功能便是通过该集合中的一个数值访问更多的一些东西。例如一个图书管理员，手中有一个库存序列号的名单。这个库存序列号除了可以判定一个给定的序列号是否在图书馆内，还能够通过该序号下面的一系列信息，快速地找到该货物存在哪个书柜、哪个书架之上。而数据库的索引便是同样的原理。例如用户在一个数字

列上建立索引，然后想要查找包含 3 的行。那么通过索引，数据库首先可以判断出 3 是否存在，然后还需要某种机制，能够把 3 所对应的行都找到，然后在表中取得这些行的其他列信息并返回给用户。

那么怎么将索引中的数据对应到相应的行呢？从图书管理员的角度看，他的名单中一定要有序列号对应的书架和书柜。那么数据库同样也有类似的东西，需要通过某种数据结构找到对应一个索引数值的所有数据，包括它们所在的数据页与数据页中的位置。这个数据结构叫作 Record ID，简称 RID。

每一个 RID 包含着数据页与 solt（也就是数据页中的逻辑位置，每一个数据页中都有一组列表将 slot 转换成真实的偏移地址）。通过访问一个 RID，数据库就可以知道一个给定的 RID 所对应的唯一的行。

此时的索引结构就相对清晰了。通过 B+树可以将键值（搜索的数据）迅速定位到叶节点，而在叶节点中，每一个键值对应着一组 RID，每一个 RID 对应表中的一行数据。

来看下面的例子，假设有一个 User 表，如表 7.1 所示。

表 7.1　User 表

User ID	Name	Salary	Phone
2	Jim	8500	212345
78	Marry	9800	31323300
5	Tom	4500	65890088
9	Jason	6300	56789000
100	Jake	1200	82631234
40	David	7800	96960033
3	Tod	10200	12347890
27	Bob	5900	45452134
65	Lucy	8700	66560066

如果在表 7.1 中有一个 User ID 的索引，可以想象成图 7.10 所示的样式。

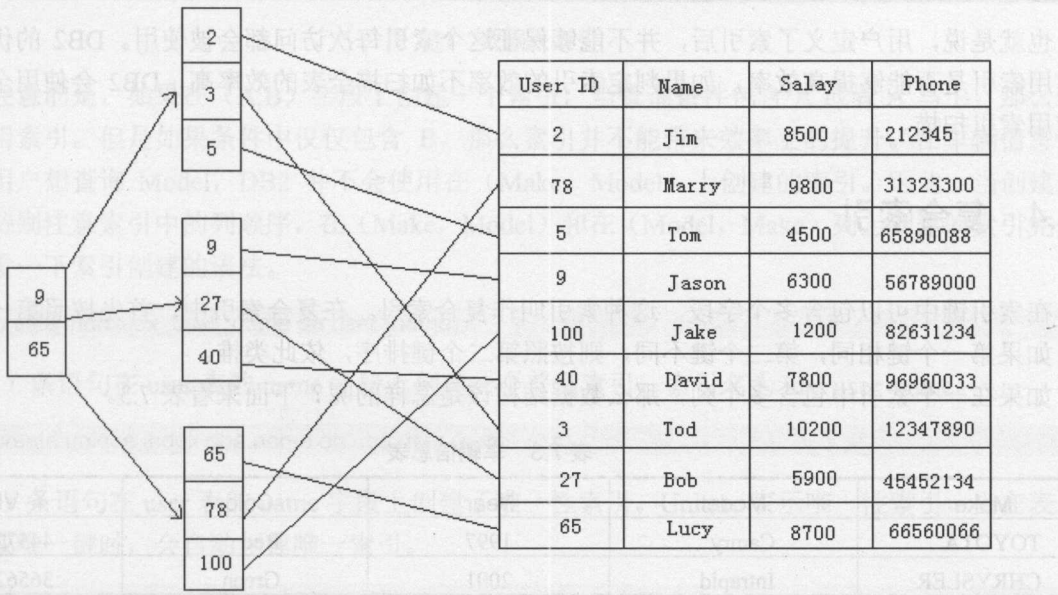


图 7.10　User ID 索引的逻辑结构

通过这个索引，假设用户查找一个 User ID 为 40 的人的电话，则需要在索引的根节点中找到中间指针，然后读取叶节点找到 40，接着根据 40 所包含的 RID 找到对应的页面与偏移位置，快速得到该行用户数据。

如果这个表中包含 100 万行或更多行数据，在没有索引的情况下，要查找 User ID=40，需要扫描整个表。但是当索引存在时，只需要读取几个索引数据页就可以找到 RID，然后访问相应的数据页面读取数据，大大减少了 I/O 读取次数。

7.4.3 合适的索引

那么什么是一个合适的索引呢？当判断一个索引能否被使用时，一方面应该看这个 B+树所包含的数值，如表 7.1 所示，如果用户想要通过 Name 查找，那么使用 UserID 定义索引则毫无意义，因为该索引中每一个节点都为 User ID=5，可考虑在 Name 列创建另外一个索引。

如何判断一个索引能否过滤大部分数据呢？以表 7.2 为例，同样在 User ID 上面创建索引，该索引中仅包含一个键值，而该键值中包含 4 个 RID 指向这个表中所有的行。这时 DB2 可能不会选择索引，因为扫描索引的代价比扫描表的代价要大。

表 7.2 相同 User ID 的 User 表

User ID	Name	Salary	Phone
5	Jim	8500	212345
5	Marry	9800	31323300
5	Tom	4500	65890088
5	Jason	6300	56789000
5	Jake	1200	82631234

也就是说，用户定义了索引后，并不能够保证这个索引每次访问都会被使用。DB2 的优化器会判定使用索引是否能够提高效率，如果判定索引的效率不如扫描全表的效率高，DB2 会使用全表扫描而不使用索引扫描。

7.4.4 复合索引

在索引键中可以包含多个字段，这种索引叫作复合索引。在复合索引中，首先按照第一个键值排序，如果第一个键相同，第二个键不同，则按照第二个键排序，依此类推。

如果在一个索引中包含多个列，那么数据结构该是怎样的呢？下面来看表 7.3。

表 7.3 车辆信息表

Make	Model	Year	Color	VIN
TOYOTA	Camry	1997	Red	445723
CHRYSLER	Intrapid	2001	Green	36562236
TOYOTA	Corolla	2011	Black	78965632

续表

Make	Model	Year	Color	VIN
TOYOTA	Corolla	2010	Gray	856236563
HONDA	Civic	1997	Silver	789652653
CHRYSLER	Intrapid	2011	Silver	785455211
HONDA	Accord	2010	Red	554256988
HONDA	Prelude	2012	Red	452631232
HONDA	Accord	2013	Black	123656326

如果在 Make 与 Model 字段创建索引，那么索引的结构如图 7.11 所示。首先按照 Make（制造商）排序，当 Make 相同时，再按照 Model 排序，如 Honda Civic 就排在 Honda Accord 之后。

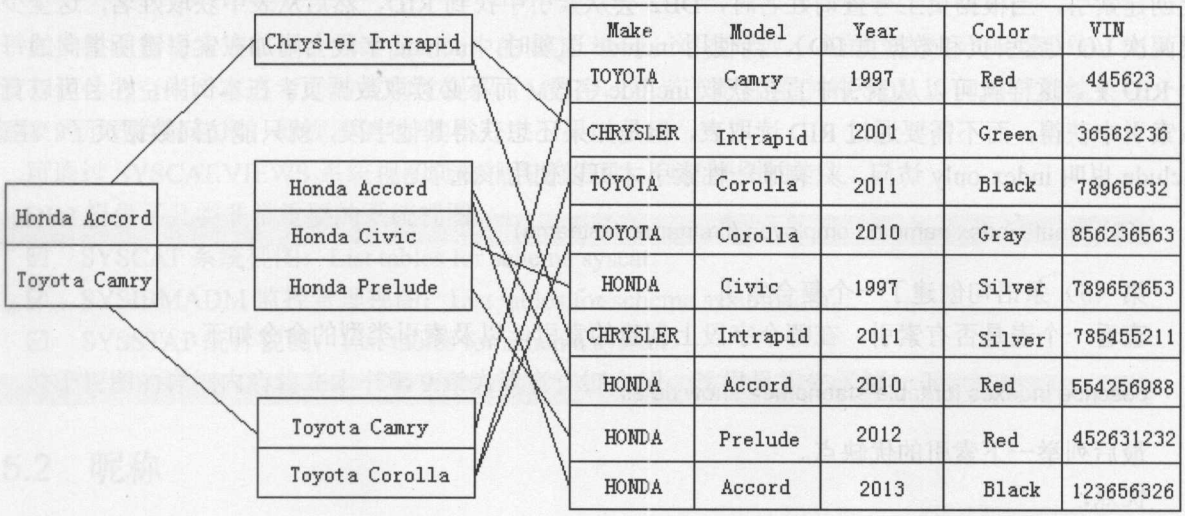


图 7.11　数据表索引

需要注意的是，如果在（A,B）字段上创建一个索引，当查询条件包含 A 或者 A 与 B，那么可以有效地使用索引。但是如果条件中仅仅包含 B，那么索引并不能带来效率上的提升。在车辆信息例子中，如果用户想查询 Model，DB2 并不会使用在（Make，Model）上创建的索引。因此，当创建索引时一定要特别注意索引中的列顺序，在（Make，Model）和在（Model，Make）列上创建的索引完全不同。下面看一下索引创建的语法。

```
（1）create index idx_user_name on user (name)
```

第（1）条语句在 user 表的 name 字段上创建一个普通索引，索引名为 idx_user_name。

```
（2）create unique index dba.name on dba.user (name asc)
```

第（2）条语句在 user 表的 name 字段上创建了唯一性索引。Unique 表示唯一性索引，当在表上创建了主键或唯一键时，会自动创建唯一索引。

```
（3）create index item on stock (itemno) cluster
```

第（3）条语句指定了 Cluster 选项，Cluster 表示集群或簇的意思，目的是尽量使数据页的物理顺序和索引键顺序保持一致。默认情况下，表数据的物理组织是无序的，数据可能杂乱无章地分散在很多数据页中，这样当按照某个范围查询一组数据时，就要根据索引叶子节点的 RID 映射到很多页中获取数据，如果数据量很大，需要的 I/O 也是很大的。如果这些数据在物理上是连续的，那么叶子节点的 RID 可能指向相邻的数据页，这时数据的获取就更快捷，需要的页数也更少，效率更高。这就是 Cluster 索引的目的。

```
（4）create unique index empidx on employee (empno) include (lastname,firstname)
```

第（4）条语句指定了 include 选项。有些读者可能不熟悉该选项，其实在某些场景，include 对效率的提升也很明显。例如，对于 employee 表来说，最经常访问的字段是员工号和姓名，如果在员工号上创建索引，当根据员工号查询姓名时，DB2 会从索引中找到 RID，然后从表中获取姓名，这至少需要两次 I/O（索引页和数据页 I/O）。当使用 include 选项时，include 字段会附加在索引键所指向的每一个 RID 上，这样就可以从索引中直接获取 include 字段，而不必读取数据页。在本例中，姓名可以直接从索引中获得，而不需要通过 RID 读取表，但是如果还想获得其他字段，就只能访问数据页了。注意：include 也叫 index-only 访问，只有唯一性索引才可以使用该选项。

```
（5）create index name on employee (firstname, lastname)
```

第（5）条语句创建了一个复合索引。

查看一个表是否有索引、在哪个字段上创建的索引，以及索引类型的命令如下：

```
describe indexes for table <tabname> show detail
```

最后列举一下索引的优缺点。

优点：

- ☑ 避免不必要的表扫描。
- ☑ 避免排序操作。
- ☑ 确保唯一性。
- ☑ 加快查询速度。
- ☑ 减少死锁发生的概率。
- ☑ 提供 cluster 顺序。

缺点：

- ☑ 增加 insert、update、delete 等操作的负担。
- ☑ 需要额外的磁盘空间。

7.5 其他常用数据库对象

其他数据库对象包括：视图、昵称、序列和自增字段。下面分别进行介绍。

7.5.1　视图

视图（View）是一个逻辑表，可以把视图想象成一个复杂的 SQL 语句的别名，当查询视图时，就是执行其中的 SQL 语句，也就是查询基础表。视图在数据库中是没有存储空间的，基础表的数据更改了，视图的结果也会随之改变。视图的好处体现在：简化用户数据查询和处理操作，隐藏数据复杂性，通过限制基表的行或列提供安全权限控制。但在应用开发设计时，不建议使用过大的视图，特别是嵌套视图，当遇到 SQL 性能问题时，包含大量视图的 SQL 调优是个梦魇。

创建视图的 SQL 语法如下：

```
CREATE VIEW <视图名> （列组名） AS <子查询>
```

如果视图查询的效率低，应该优化其中的 SQL 语句。对实时性不高的报表或分析型系统，如果实在没有优化的空间，可以考虑做成物化视图（MQT），相当于把视图中的 SQL 语句的查询结果以一个物理表的形式存储起来，即视图的物理化。

可通过 SYSCAT.VIEWS 系统视图查看视图的信息，包括视图的定义。

DB2 提供了几类非常重要的系统视图。

☑ SYSCAT 系统视图：List tables for schema syscat。

☑ SYSIBMADM 监控管理视图：List tables for schema sysibmadm。

☑ SYSSTAT 统计视图：List tables for schema sysstat。

关于视图的详细内容将在本书第 9 章为读者详细介绍，这里只需先了解一下。

7.5.2　昵称

昵称是 DB2 联邦数据库中的概念，正常情况下，在一个数据库中访问另外一个数据库中的表是不允许的，但是配置了联邦后，在本地为远程表创建一个昵称，就可以访问远端数据库了。关于 DB2 联邦数据库中的内容，我们将在以后出版的书籍中详细介绍。这里只做简单说明。

7.5.3　序列

对于一些交易系统，经常需要在表中记录操作日志、交易流水等信息，通过日志号、交易流水号等标识每行记录，那么如何保证记录的唯一性呢？一种通用的设计方法是为每行记录存取一个计数器，当某个应用需要插入记录时首先获取表中最大的计数器，加 1，然后提交。不幸的是，这种方法在处理多用户并发情况下存在性能问题。在数据库级别，DB2 提供了如下几种方法解决字段唯一问题：

☑ Genearate_unique 函数。

☑ Identity 标识字段。

☑ 序列（Sequence）对象。

Generate_unique 函数是 DB2 最早引入的在表中生成唯一值的方法，该函数的返回值是国际标准时

间（UTC）生成的当前时间戳加上当前数据库分区号，包含 13 个字节的数据串（char（3）for bit data）。但需要注意的是：当前时间戳是根据当前系统时间生成的，当调整了系统日期，可能会出现重复值的情况。

序列（Sequence）是 DB2 的一个对象，按照一定规则产生自动增加的数字序列。这个序列一般用作主键，没有其他意义。通过 NEXTVAL FOR<seqname>表达式产生序列的下一个值，通过 PREVVAL 返回序列的前一个值。Sequence 的语法如下：

```
CREATE SEQUENCE seq_name
START WITH A
INCREMENT BY B
MINVALUE C
MAXVALUE D
NO CYCLE
CACHE E
ORDER
```

其中各参数如下。

- ☑ OR REPLACE：若有创建序列重名的序列，那么将会替代（先删除，再创建）。
- ☑ AS DATATYPE：指定数据类型，这里说明一下，序列的数据类型只能为数值型，如 smallint、integer、bigint，不带小数点的 decimal 类型。
- ☑ START WITH：指定序列的起始值，默认情况下，对于升序的序列是当前指定数据类型的最小值，对于降序序列是当前指定类型的最大值。
- ☑ INCREMENT BY：指定增长的值，默认值为 1，正数表示此序列为增长升序的，负数表示此序列为降序的。
- ☑ MINVALUE：最小值，若降序序列，no cycle 的话，到此值时就不再生成序列值了。
- ☑ NO MINVALUE：对于升序序列来说，此值为 start with 的值，如果 start with 值未指定的话就是 1，对于降序序列来说，此值就是指定数据类型的最小值。
- ☑ MAXVALUE：指定生成序列的最大值。
- ☑ NO MAXVALUE：对于升序序列来说，此值为指定数据类型的最大值；对于降序序列来说，此值为 start with 值，若未指定 start with 值的话就是-1。
- ☑ CYCLE：循环使用数据值，对于升序序列来说，当达到了最大值之后，下一个值将会是其最小值；对于降序序列来说，达到最小值后，下一个序列值为其最大值。
- ☑ NO CYCLE：当达到序列的边界值之后，就不再产生序列值，默认选项。
- ☑ CACHE：缓存序列值，表示每次应用此序列时，预先生产并存放在内存中的序列值。其作用是有效降低了写日志的 I/O 操作。若在使用的过程中，出现系统错误的话，那么所有这些缓存值将会丢失。最小值为 2，默认为 20。
- ☑ NO CACHE：当指定此选项时，内存中不会存储任何序列值，无论出现什么异常现象都不会影响到此序列，每次生成新的序列值，都会导致写日志的 I/O 操作。
- ☑ ORDER：按照请求的顺序生成值。
- ☑ NO ORDER：不会按照请求的顺序生成值，默认情况。

下面创建一个序列：

```
CREATE SEQUENCE db2inst1.my_seq
    AS bigint
    START WHTH 1
    INCREMENT BY 1
    NO MAXVALUE
    CYCLE
    CACHE 100;
```

序列的用法相对简单，需要注意的是 cache 选项，cache 的目的是将一组连续值先计算出来，缓存到内存中，每次使用时从内存中直接获取即可，当这组值用完时再分配一组。这些缓存值对所有连接是共享的，如果所有连接断开，则缓存的值将丢失，下次启动数据库时会从下一组值分配，因此当 cache 不等于 1 时，可能会造成序列的不连续。

序列对象会因为各种理由而用于应用程序中。这样做有如下好处：

☑ 无须等待值的增加。

☑ 独立的连续数字生成对象（不与表关联）。

☑ 递增或递减地生成数字。

☑ 用于从其他数据库移植应用程序。

☑ 可以生成键，用以跨多个表（RI 或相关的应用程序）协调各键。

7.5.4 自增字段

Indentity 的作用也是用来生成自增数字序列的。Indentity 的定义有两种方法，一种是 generated always as identity，表示自增列值由系统自动产生，用户不能指定；另外一种通过 generated by default as identity，是指当用户提供了值时，自增字段的值按照用户提供的值进行插入，当用户没有提供自增字段的值时，系统会自动生成。

下面来看这两种方式分别对应的两个示例。

（1）使用 generated always as identity：

```
CREATE TABLE demo1(
DEMONO BIGINT NOT NULL GENERATED ALWAYS AS IDENTITY(
            START WITH 500,                          --自增主键的起始值
            INCREMENT BY 1,                          --每次主键生成的增量
MINVALUE 500,                                        --最小值
MAXVALUE 9223372036854775807,                        --主键最大值
NO CYCLE,                                            --是否可循环
NO CACHE,                                            --是否缓存
NO ORDER
),
NAME VARCHAR(16))
```

```
INSERT INTO demo1 (name) VALUES ('zhaoming')                    --允许插入
INSERT INTO demo1 (demono,name) VALUES (502,'zhaoming')         --不允许插入
```

（2）使用 generated by default as identity：

```
CREATE TABLE demo2(
DEMONO BIGINT NOT NULL GENERATED BY DEFAULT AS IDENTITY(
        START WITH 500,                                         --自增主键的起始值
        INCREMENT BY 1,                                         --每次主键生成的增量
MINVALUE 500,                                                   --最小值
MAXVALUE 9223372036854775807,                                   --主键最大值。
NO CYCLE,                                                       --是否可循环
NO CACHE,                                                       --是否缓存
NO ORDER
),
NAME VARCHAR(16))
INSERT INTO demo2 (name) VALUES ('zhaoming')                    --允许插入
INSERT INTO demo2 (demono,name) VALUES (502, 'zhaoming')        --允许插入
```

可以使用 SYSCAT.COLUMNS 视图找出数据库中所有带自增列的表和字段，INDENTITY 字段用来标识是否为自增列，generated 字段值为‘A’表示 generated always 类型，‘D’表示 generated by default 类型。在命令窗口中编写如下命令：

```
db2 CONNECT TO mrkj USER mr USING 1                             --首先连接数据库
--回车
db2 SELECT substr(tabname,1,20) as tabname, substr(colname,1,20) as colname, identity,
generated FROM SYSCAT.COLUMNS WHERE IDENTITY = 'Y'
--回车
```

显示结果如图 7.12 所示。

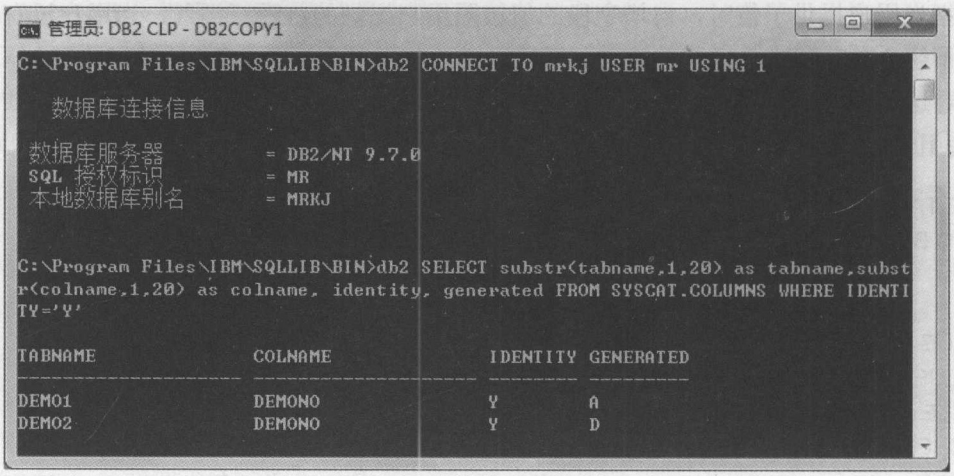


图 7.12　找出数据库中带自增列的表和字段

使用 Sequence 对象代替 Identity 列有许多好处，表 7.4 是两者的简单比较。

表 7.4　Sequence 对象与 Identity 列之间的比较

Sequence 对象	Identity 列
单独的 Sequence 对象是在用户请求时创建的	由 DB2 生成/维护和填充的内部 Sequence
可用于它们所选择的任何目的，并且存在多个	与特定的表相关联，并且只能有一个
循环（CYCLE）将回绕和重复，并且无须进行唯一性考虑	如果唯一索引在 Identity 列上并且创建了复制，那么循环（CYCLE）可能会产生问题
当用于填充表时，后来可被更新	如果是 GENERATED ALWAYS，就无法被更新
可以更改（ALTER）Sequence 对象的属性。还可以进行注释（COMMENT）以及授予/撤销（GRANT/REVOKE）权限	只能更改表（ALTER TABLE）（如果向已填充的表添加 Identity 列，将被置于 REORG 暂挂状态）
可被删除	不能从表中删除
支持 NEXT VALUE FOR EXPRESSION 和 PREVIOUS VALUE FOR EXPRESSION	必须使用 ID_VAL_LOCAL 并且只返回用户提交（commit）范围内的最后值

注意

在今后的设计中，如果使用 Sequence 对象比使用 Identity 列更好，那么在选择 Identity 列时就要考虑仔细。如果它们是在已填充的表上定义的并且需要被删除，那么必须删除表然后重新创建。这对于高效环境中的大型表可能是一个大问题。

Sequence 和 Identity 的差别可归纳为：Sequence 是数据库系统中的一个对象，可以在整个数据库中使用，和表没有任何关系；而 Indentity 必须指定在表中某一列，作用范围就是这个表。在应用设计中，要根据具体情况来选择使用哪一种。

Sequence 用于下列情况中：

- ☑ 要将从一个序列生成的值存储到多个表中。
- ☑ 每个表有多个列需要自动生成的值（可能通过使用同一个序列或多个序列为每一行生成多个值）。
- ☑ 生成新值的进程与任何表的引用无关。

Identity 用于下列情况中：

- ☑ 表中只有一列需要自动生成的值。
- ☑ 每一行都需要独立的值。
- ☑ 用自动生成器来生成表的主键。
- ☑ 生成新值的进程与对表的插入操作紧密联系，无论插入操作是如何发生的。

7.6　小　　结

本章主要讨论了 DB2 中的各种数据对象。熟练使用各种类型的对象，以及了解它们的结构和原理，在数据库维护中会给用户带来极大的方便。现在很多的 DBA 都已经或多或少地参与了应用程序的开

发，拥有数据对象原理的知识，DBA 可以在应用程序的开发阶段对开发人员提出很多宝贵意见，减少后期修改的时间，也减少系统的维护成本。

7.7 实践与练习

（1）什么是模式？（**答案位置：光盘\TM\sl\7\2**）

（2）表约束有哪些？都是做什么的？（**答案位置：光盘\TM\sl\7\3**）

（3）举例说明索引的好处。（**答案位置：光盘\TM\sl\7\4**）

第2篇

核心技术

- 第 8 章　DB2 数据查询技术
- 第 9 章　视图的使用
- 第 10 章　约束、索引和别名
- 第 11 章　SQL 过程语言
- 第 12 章　DB2 中的函数
- 第 13 章　存储过程和触发器

本篇介绍 DB2 数据查询技术、视图的使用、约束、索引和别名、SQL 过程语言、DB2 中的函数、存储过程和触发器等内容。学习完这一部分，读者能够对 DB2 数据库进行基本的日常管理和维护。

第 8 章

DB2 数据查询技术

（视频讲解：78 分钟）

本章将介绍 DB2 语言的基本查询，包括简单查询、筛选、分组、排序、多表关联查询和使用聚集函数计算列值等，学习这些内容可以熟悉 SQL 语言，掌握 DB2 SQL 的基础，有助于进一步的学习和程序开发。

通过阅读本章，您可以：

- 了解选择、投影、连接和集合运算
- 掌握检索数据操作
- 掌握子查询的用法

在数据库应用中，最常用的操作是查询，查询是数据库其他操作（如插入、修改和删除）的基础。在 DB2 中对数据库的查询使用 SELECT 语句。SELECT 语句的功能非常强大，本章将讨论用 SELECT 语句对数据库进行的各种查询。

8.1　选择、投影、连接和集合运算

关系数据库建立在关系模型基础之上，具有严格的数学理论基础。关系数据库对数据的操作定义了一组关系运算：选择、投影和连接。

8.1.1　选择

选择（SELECT）是单目运算，其中运算的对象是表。此运算是按照指定的条件，从表中选择满足条件的行形成一个新表来作为运算结果的。

选择运算的符号为：σ_F（R）。其中 σ 是选择运算符，下标 F 是一个条件表达式，R 是被操作的表。

【例 8.1】　在“学生表”中查找出总分数为 88 的学生的行形成一个新表，如图 8.1 所示（**实例位置：光盘\TM\sl\8\1**）。

学号	姓名	专业	性别	出生日期	总分数	照片	联系方式	备注
14001	马芳	计算机应用技术 ...	女	1989-8-16	89		XML ...	
14002	宋静	计算机应用技术 ...	女	1990-5-12	83		XML ...	
14003	周天星	计算机应用技术 ...	男	1989-2-17	92		XML ...	三好学生，已提...
14004	蒋阳泉	计算机应用技术 ...	男	1991-7-13	84		XML ...	
14005	董敏	计算机应用技术 ...	女	1992-8-5	88		XML ...	

图 8.1　学生表

运算式为：σ_F（学生表），F：总分数=88。

翻译成 SQL 语句是：

```
SELECT * FROM 学生表 WHERE 总分数=88
```

注意

SELECT FROM 语句会在本章 8.2 节中为大家详细讲解。在查询数据前必须先连接到被查询的数据库。本章所有 SQL 语句连接的都是 MRKJ 数据库。

该运算结果如图 8.2 所示。

学号	姓名	专业	性别	出生日期	总分数	照片	联系方式	备注
14005	董敏	计算机应用技术 ...	女	1992-8-5	88		XML ...	

图 8.2　满足条件的学生信息

8.1.2 投影

投影（PROJECTION）也是单目运算，该运算从表中选出指定的属性组成一个新表。

投影运算的符号为：Π_A（R）。其中 Π 是投影运算符，下标 A 是属性名（即表中的列名），R 是被操作的表名。

【例 8.2】 在图 8.1 中对“学号”、“姓名”和“总分数”进行投影操作（**实例位置：光盘\TM\sl\8\2**）。

运算式为：$\Pi_{学号,姓名,总分数}$（学生表）。

翻译成 SQL 语句为：

```
SELECT 学号,姓名,总分数 FROM 学生表
```

该运算结果如图 8.3 所示。

学号	姓名	总分数
14001	马芳	89
14002	宋静	83
14003	周天星	92
14004	蒋阳泉	84
14005	董敏	88

图 8.3 使用投影运算结果

8.1.3 连接

连接（JOIN）是根据数据的某些公共领域对来自两个或更多个表的数据进行组合的过程。

连接运算的 SQL 语法如下：

```
SELECT <列名> FROM <表 1> JOIN <表 2> ON <条件>
```

【例 8.3】 将“学生表”和“成绩表”执行连接运算，如图 8.4 和图 8.5 所示（**实例位置：光盘\TM\sl\8\3**）。

学号	姓名	专业	性别	出生日期	总分数	照片	联系方式	备注
14001	马芳	计算机应用技术 ...	女	1989-8-16	89		XML ...	
14002	宋静	计算机应用技术 ...	女	1990-5-12	83		XML ...	
14003	周天星	计算机应用技术 ...	男	1989-2-17	92		XML ...	三好学生，已提...
14004	蒋阳泉	计算机应用技术 ...	男	1991-7-13	84		XML ...	
14005	董敏	计算机应用技术 ...	女	1992-8-5	88		XML ...	

图 8.4 学生表

在“命令编辑器”中输入如下命令，查询结果如图 8.6 所示。

```
--a 和 b 是自定义的表别名
SELECT  a.学号,a.姓名,b.课程号,b.成绩 FROM 学生表 a JOIN 成绩表 b ON a.学号=b.学号
```

学号	课程号	成绩
14001	101	89
14002	101	83
14003	101	92
14004	101	84
14005	101	88

图 8.5　成绩表

学号	姓名	课程号	成绩
14001	马芳	101	89
14002	宋静	101	83
14003	周天星	101	92
14004	蒋阳泉	101	84
14005	董敏	101	88

图 8.6　使用 JOIN 连接查询

两个表连接最常用的条件就是两个表的某列值相等，这样的连接称为等值连接，上面的例子就是等值连接。

在 DB2 中，默认的连接方式是内连接（INNER JOIN），对应的其他方式称为外连接（OUTER JOIN），两种方式的主要区别是：

☑　内连接只显示满足连接条件的数据集。

☑　外连接以指定表为连接主体，将主体表中不满足连接条件的数据集也显示出来。

针对之前的内连接查询，使用 WHERE 子句代替 ON 子句也可查询到同样的结果：

```
SELECT  a.学号,a.姓名,b.课程号,b.成绩 FROM 学生表 a , 成绩表 b
WHERE a.学号=b.学号
```

虽然可以得到相同的结果，但是我们并不能使用 WHERE 子句替代 ON 子句的功能。因为两种方式实现的机制是不一样的，而且 WHERE 子句并不全适用于外连接的情况。为了防止查询语句书写混乱，我们需要遵循如下原则：连接添加使用 ON 子句，过滤条件使用 WHERE 子句，桥梁功能和过滤功能不要混合使用。

8.1.4　集合运算

比起连接查询，集合运算主要用来处理数据之间的并（Union）、差（Difference）和交（Intersection）操作，相对应的 SQL 运算符为 UNION（合集）、EXCEPT（差集）、INTERSECT（交集）和 MINUS（减集）。

基本语法如下：

```
                UNION
                UNION ALL
                INTERSECT
<SELECT 表达式>  INTERSECT ALL          <SELECT 表达式>
```

```
EXCEPT
EXCEPT ALL
MINUS
```

在这 7 种运算符中，UNION 和 UNION ALL 是使用最多的。

【例 8.4】 在 MRKJ 数据库中创建一个名称叫“新成绩表”的表，“新成绩表”中的列与“成绩表”中的列相同，然后合并“新成绩表”和“成绩表”中的所有数据（**实例位置：光盘\TM\sl\8\4**）。

```
SELECT * FROM 成绩表 UNION ALL SELECT * FROM 新成绩表
```

本例运行结果如图 8.7 所示。

不难发现，合并后的表中有重复数据，这是因为使用 UNION ALL 运算符，DB2 会对两张表中的所有数据做简单的合并。但是一般情况下我们不需要显示重复的数据行，这时就可以使用 UNION 来合并数据了。

使用 UNION 合并数据的命令如下：

```
SELECT * FROM 成绩表 UNION SELECT * FROM 新成绩表
```

合并后的结果如图 8.8 所示。

学号	课程号	成绩
14001	101	89
14002	101	83
14003	101	92
14004	101	84
14005	101	88
14006	101	63
14007	101	61
14008	101	69
14001	101	89
14002	101	83
14003	101	92
14004	101	84
14005	101	88

图 8.7 使用 UNION ALL 合并数据

学号	课程号	成绩
14008	101	69
14007	101	61
14006	101	63
14005	101	88
14004	101	84
14003	101	92
14002	101	83
14001	101	89

图 8.8 使用 UNION 合并数据

下面分别创建两个表 TEST1 和 TEST2，并插入表 8.1 中的数据。

表 8.1　表 TEST1 和表 TEST2 中的数据

TEST1	TEST2
A	A
A	A
A	B
B	B
B	B
C	C
D	

通过几种集合运算得到的对应结果如表 8.2 所示。

表 8.2　表 TEST1 和表 TEST2 通过集合运算后的结果

运　算　符	结　果
INTERSECT	A B C
INTERSECT ALL	A A B B C
EXCEPT	D
EXCEPT ALL	A D
MINUS	D

8.2　检 索 数 据

用户对表或视图最常进行的操作就是检索数据，检索数据可以通过 SELECT 语句来实现，该语句由多个子句组成，通过这些子句可以完成筛选、投影和连接（投影和连接在 8.1 节已经讲过，这里不再赘述）等各种数据操作，最终得到用户想要的查询结果。检索数据的基本语法格式由 9 个子句组成：

```
<select {[distinct | all ] columns | *}>
[into table_name]
<from {tables | views | other select}>
[where conditions]
[group by columns]
[having conditions]
[order by columns]
[fetch first class]
[union <select query clause>]
```

在上面的语法中共有 9 个子句，它们的功能分别如下。

- ☑ select 子句：用于选择数据表、视图中的列。
- ☑ into 子句：用于将原表的结构和数据插入到新表中。
- ☑ from 子句：用于指定数据来源，包括表、视图和其他 select 语句。
- ☑ where 子句：用于对检索的数据进行筛选。
- ☑ group by 子句：用于对检索结果进行分组显示。
- ☑ having 子句：用于从使用 group by 子句分组后的查询结果中筛选数据行。
- ☑ order by：用于对结果集进行排序（包括升序和降序）。
- ☑ fetch first：用于返回结果集的行数。
- ☑ union：用于连接两条 select 查询语句。

说明

“<>”表示该子句是必选的，查询语句必须包含 SELECT 子句和 FROM 子句。“[]”表示该子句是可选的。SELECT 语句的强大力量就在于它能够使用无穷多个变量（使用有限的规则集）来构建复杂的查询，它可以很复杂。SELECT 动词是可以递归的，这意味着：SQL 语句的输入可以是后续的一连串嵌套 SQL 语句的输出。例如，嵌套的 SELECT 子查询可以被嵌套在其他的数据操作语言语句中。

接下来将对上面的各种子句和查询方式进行详细介绍。

8.2.1 简单查询

只包含 SELECT 子句和 FROM 子句的查询就是简单查询，SELECT 子句和 FROM 子句是 SELECT 语句的必选项，也就是说每个 SELECT 语句都必须包含这两个子句。其中，SELECT 子句用于选择想要在查询结果中显示的列，对于这些要显示的列，既可以使用列名来表示，也可以使用星号（*）来表示。在检索数据时，数据将按照 SELECT 子句后面指定的列名的顺序来显示；如果使用星号（*），则表示检索所有的列，这时数据将按照表结构的自然顺序来显示。

1. 检索所有的列

如果想要检索指定数据表的所有列，可以在 SELECT 子句后面使用星号（*）来实现。检索指定数据表时，首先要连接数据库。

语法为：

```
SELECT * FROM 表名;  --如果要查询表中所有列，SELECT 后面就加星号（*）
```

【例 8.5】 使用星号（*）检索 MRKJ 数据库中学生表中的数据（**实例位置：光盘\TM\sl\8\5**）。

```
SELECT * FROM 学生表;
```

本例运行结果如图 8.9 所示。

学号	姓名	专业	性别	出生日期	总分数	照片	联系方式	备注
14001	马芳	计算机应用技术 ...	女	1989-8-16	89		XML ...	
14002	宋静	计算机应用技术 ...	女	1990-5-12	83		XML ...	
14003	周天星	计算机应用技术 ...	男	1989-2-17	92		XML ...	三好学生，已提...
14004	蒋阳泉	计算机应用技术 ...	男	1991-7-13	84		XML ...	
14005	董敏	计算机应用技术 ...	女	1992-8-5	88		XML ...	

图 8.9　检索“学生表”中所有数据

在上面的例子中，FROM 子句的后面只有一个数据表，实际上可以在 FROM 子句的后面指定多个数据表，每个数据表名之间使用逗号（,）分隔开，其语法格式如下：

```
SELECT * FROM table_name1, table_name2, table_name3……table_namen
```

【例 8.6】　在 FROM 子句中指定两个数据表（**实例位置：光盘\TM\sl\8\6**）。

```
select * from 成绩表,学生表;
```

注意

在本书第 6 章中介绍过 SQL 关键字不区分大小写，也就是说怎么写都可以但不要写错，SQL 中的符号一定要在英文状态下输入。本书中有两种书写格式，一种是：mr.表名；一种是直接写表名。例如：mr.学生表与直接写学生表是等价的。

本例运行结果如图 8.10 所示。

学号	课程号	成绩	学号1	姓名	专业	性别	出生日期	总分数	照片
14001	101	89	14001	马芳	计算机应用技术 ...	女	1989-8-16	89	
14002	101	83	14001	马芳	计算机应用技术 ...	女	1989-8-16	89	
14003	101	92	14001	马芳	计算机应用技术 ...	女	1989-8-16	89	
14004	101	84	14001	马芳	计算机应用技术 ...	女	1989-8-16	89	
14005	101	88	14001	马芳	计算机应用技术 ...	女	1989-8-16	89	
14001	101	89	14002	宋静	计算机应用技术 ...	女	1990-5-12	83	
14002	101	83	14002	宋静	计算机应用技术 ...	女	1990-5-12	83	
14003	101	92	14002	宋静	计算机应用技术 ...	女	1990-5-12	83	
14004	101	84	14002	宋静	计算机应用技术 ...	女	1990-5-12	83	
14005	101	88	14002	宋静	计算机应用技术 ...	女	1990-5-12	83	
14001	101	89	14003	周天星	计算机应用技术 ...	男	1989-2-17	92	

图 8.10　同时查询“成绩表”和“学生表”

2. 检索指定列

用户可以指定查询表中的某些列而不是全部的列，并且被指定列的顺序不受限制，指定部分列也称作投影操作。这些列名紧跟在 SELECT 关键字的后面，每个列名之间用逗号分隔开，其语法格式

如下：

```
select column_name1,column_name2,column_name3,……column_namen;
```

说明

利用 SELECT 指定列的好处就是可以改变列在查询结果中的默认显示顺序。

【例 8.7】 检索 MRKJ 数据库的“学生表”中指定的列（学号，姓名，性别，专业）（实例位置：光盘\TM\sl\8\7）。

```
select 学号,姓名,性别,专业 from 学生表;
```

注意

上面查询结果中列的显示顺序与“学生表”结构的自然顺序不同。

本例运行结果如图 8.11 所示。

学号	姓名	性别	专业
14001	马芳	女	计算机应用技术 ...
14002	宋静	女	计算机应用技术 ...
14003	周天星	男	计算机应用技术 ...
14004	蒋阳泉	男	计算机应用技术 ...
14005	董敏	女	计算机应用技术 ...

图 8.11 指定列名查询

3. 带有表达式的 SELECT 子句

在使用 SELECT 语句时，对于数字数据和日期数据都可以使用算术表达式。在 SELECT 语句中可以使用算术运算符，包括加（+）、减（-）、乘（*）、除（/）和括号。另外，在 SELECT 语句中不仅可以执行单独的数学运算，还可以执行单独的日期运算以及列名关联的运算。

【例 8.8】 检索 MRKJ 数据库的“学生表”中的“总分数”列，把其值在原值的基础上增加 1（实例位置：光盘\TM\sl\8\8）。

```
select 总分数+1,总分数 from mr.学生表;
```

注意

在例 8.8 的查询结果中，左侧显示的是“总分数”列调整为原值的基础上增加 1 后的值，右侧显示的是“总分数”列的原值。

本例运行结果如图 8.12 所示。

4．为列指定别名

通常情况下，数据库中的表名和列名都是用英文来命名的，本书中为了初学者查看方便就以中文来命名。但是在今后的操作中，会有许多数据表的列名和表名以英文的方式命名，用户为了方便查看检索结果，常常需要为这些列指定别名。在 DB2 中，使用 AS 关键字为列指定别名，也可以不使用任何关键字而直接指定。

【例 8.9】　检索 MRKJ 数据库的“学生表”指定的列（学号，姓名，总分数），并使用 AS 关键字为这些列指定英文的别名（**实例位置：光盘\TM\sl\8\9**）。

```
select 学号 AS sid, 姓名 AS name, 总分数 AS total_score
    from mr.学生表;
```

或

```
select 学号 sid, 姓名 name, 总分数 total_score
    from mr.学生表;
```

本例运行结果如图 8.13 所示。

1	总分数
90	89
84	83
93	92
85	84
89	88

图 8.12　运行结果

SID	NAME	TOTAL_SCORE
14001	马芳	89
14002	宋静	83
14003	周天星	92
14004	蒋阳泉	84
14005	董敏	88

图 8.13　为列指定别名

5．计算列标题

【例 8.10】　列标题中包含空格的查询语句（**实例位置：光盘\TM\sl\8\10**）。

```
select 学号 AS "si d", 姓名 AS "na me", 总分数 AS "total score"
    from mr.学生表;
```

或

```
select 学号 "si d", 姓名 "na me", 总分数 "total score"
    from mr.学生表;
```

本例运行结果如图 8.14 所示。

注意

当自定义的列标题中含有“空格”时，必须使用“双引号”将列标题括起来。

si d	na me	total score
14001	马芳	89
14002	宋静	83
14003	周天星	92
14004	蒋阳泉	84
14005	董敏	88
14006	万方	92

图 8.14　更改查询结果的列标题

6. 显示不重复的记录

在默认情况下，结果集中包含所有符合查询条件的数据行，这样结果集中就有可能出现重复数据。而在实际的应用中，这些重复的数据除了占据较大的显示空间外，可能不会给用户带来太多有价值的信息，这样就需要除去重复记录，保留唯一的记录即可。在 SELECT 语句中，可以使用 DISTINCT 关键字来限制在查询结果显示不重复的数据，该关键字用在 SELECT 子句的列表前面。

【例 8.11】 显示 MRKJ 数据库的“学生表”中“总分数”的列，要求显示的“总分数”记录不重复（**实例位置：光盘\TM\sl\8\11**）。

说明

在完成本实例前，首先在“学生表”中对应的位置插入一行记录。插入的信息为：14006，万方，计算机应用技术，女，1993-8-12，92。剩下的列都为默认值。插入完成后运行结果如图 8.15 所示。

学号	姓名	专业	性别	出生日期	总分数	照片	联系方式	备注
14001	马芳	计算机应用技术 ...	女	1989-8-16	89		XML ...	
14002	宋静	计算机应用技术 ...	女	1990-5-12	83		XML ...	
14003	周天星	计算机应用技术 ...	男	1989-2-17	92		XML ...	三好学生，已提...
14004	蒋阳泉	计算机应用技术 ...	男	1991-7-13	84		XML ...	
14005	董敏	计算机应用技术 ...	女	1992-8-5	88		XML ...	
14006	万方	计算机应用技术 ...	女	1993-8-12	92		XML ...	

图 8.15　往表中新插入一行数据

查看“学生表”不难发现，“总分数”列中有两条相同的数据，即 92。接下来执行去重记录的操作：

```
select distinct 总分数 from 学生表;
```

本例运行结果如图 8.16 所示。

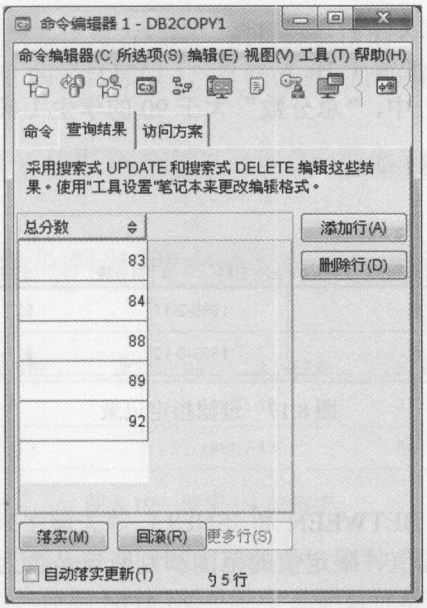


图 8.16　显示不重复的记录

8.2.2　筛选查询

在 SELECT 语句中使用 WHERE 子句可以实现对数据行的筛选操作，只有满足 WHERE 子句中判断条件的行才会显示在结果集中，而那些不满足 WHERE 子句判断条件的行则不包括在结果集中。这种筛选操作是非常有意义的，通过筛选数据，可以从大量的数据中得到用户所需要的数据。在 SELECT 语句中，WHERE 子句位于 FROM 子句之后，其语法格式如下：

```
SELECT columns_list
FROM table_name
WHERE conditional_expression
```

参数说明如下。

☑　conlumns_list：字段列表。

☑　table_name：表名。

☑　conditional_expression：筛选条件表达式。

接下来对几种常见的筛选情况进行详细讲解。

1. 比较筛选

可以在 WHERE 子句中使用比较运算符来筛选数据，这样只有满足筛选条件的数据行才会被检查出来，不满足条件的数据行则不会被检索出来。基本的“比较筛选”操作主要有以下 6 种情况。

☑　A=B：比较 A 与 B 是否相等。

☑　A!=B 或 A<>B：比较 A 与 B 是否不相等。

☑　A>B：比较 A 是否大于 B。

☑　A<B：比较 A 是否小于 B。

☑ A>=B：比较 A 是否大于或等于 B。

☑ A<=B：比较 A 是否小于或等于 B。

【例 8.12】 查询“学生表”中，“总分数”大于 90 的学生（**实例位置：光盘\TM\sl\8\12**）。

```
select * from 学生表 where 总分数>90;
```

本例运行结果如图 8.17 所示。

学号	姓名	专业	性别	出生日期	总分数	照片	联系方式	备注
14003	周天星	计算机应用技术 ...	男	1989-2-17	92		XML ...	三好学生，已提...
14006	万方	计算机应用技术 ...	女	1993-8-12	92		XML ...	

图 8.17 过滤指定记录

2. 使用特殊关键字筛选

SQL 语言提供了 LIKE、IN、BETWEEN 和 ISNULL 等关键字来筛选匹配的数据，这些关键字的功能分别是匹配字符串、查询目标值、限定值的范围和判断值是否为空等，下面将对这些关键字进行详细讲解。

（1）LIKE 关键字：在 WHERE 子句中使用 LIKE 关键字查询数据的方式也称为字符串模式匹配或字符串模糊查询。LIKE 关键字需要使用通配符在字符串内查找指定的模式，所以需要了解常用的通配符。常用的通配符有下划线“_”，它代表任意一个字符；百分号“%”，它代表任意数量的字符。例如，“K%”表示以字母 K 开头的任意长度的字符串，“%M%”表示包含字母 M 的任意长度的字符串，“_MRKJ”表示 5 个字符长度且后面 4 个字符是 MRKJ 的字符串。

【例 8.13】 使用 LIKE 关键字在“学生表”中查询姓马的学生（**实例位置：光盘\TM\sl\8\13**）。

```
select * from 学生表 where 姓名 like '马%';
```

本例运行结果如图 8.18 所示。

学号	姓名	专业	性别	出生日期	总分数	照片	联系方式	备注
14001	马芳	计算机应用技术 ...	女	1989-8-16	89		XML ...	

图 8.18 使用 LIKE 关键字

说明

可以在 LIKE 关键字的前面加上 NOT，表示否定的判断，如果 LIKE 为真，则 NOT LIKE 为假。另外，也可以在 IN、BETWEEN、ISNULL 和 ISNAN 等关键字前面加上 NOT 来表示否定的判断。

注意

使用通配符时，一定要保证它们放在模式字符串中的合适位置。在例 8.13 中，只有以“马”开头的姓名才被返回。如果指定的模式是‘%王%’，那么，只要姓名中包含了“王”的学生记录都被返回。

必须要小心地在模式字符串中使用大小写。DB2 数据库是以大小写敏感的方式存储，在模式字符串中使用的字符必须匹配要搜索的列中数据的大小写。

虽然 LIKE 关键字使用简便，但是仍需慎用。因为，处理 LIKE 关键字的速度很慢，非常耗费资源，故而效率不高。因为这个原因，只有在没有其他方法来定位所需数据时，才使用 LIKE 关键字。

（2）IN 关键字：当测试一个数据值是否匹配一组目标值中的一个时，通常使用 IN 关键字来指定列表搜索条件。IN 关键字的格式是 IN（目标值 1，目标值 2，目标值 3……），目标值的项目之间必须使用逗号分隔，并且括在括号中。

【例 8.14】 在“学生表”中，使用 IN 关键字查询总分数为（83,84,99）中任意一种的学生信息（**实例位置：光盘\TM\sl\8\14**）。

```
select * from 学生表 where 总分数 in (83,84,99);
```

本例运行结果如图 8.19 所示。

学号	姓名	专业	性别	出生日期	总分数	照片	联系方式	备注
14002	宋静	计算机应用技术 ...	女	1990-5-12	83		XML ...	
14004	蒋阳泉	计算机应用技术 ...	男	1991-7-13	84		XML ...	

图 8.19　使用 IN 关键字

另外，NOT IN 表示查询指定的值不在某一组目标值中，这种方式在实际应用中也很常见。

【例 8.15】 在“学生表”中，使用 NOT IN 关键字查询总分数不在指定目标列表（83,84,89）范围内的学生信息（**实例位置：光盘\TM\sl\8\15**）。

```
select * from 学生表 where 总分数 not in (83,84,89);
```

本例运行结果如图 8.20 所示。

学号	姓名	专业	性别	出生日期	总分数	照片	联系方式	备注
14003	周天星	计算机应用技术 ...	男	1989-2-17	92		XML ...	三好学生，已提...
14005	董敏	计算机应用技术 ...	女	1992-8-5	88		XML ...	
14006	万方	计算机应用技术 ...	女	1993-8-12	92		XML ...	

图 8.20　使用 NOT IN 关键字

（3）BETWEEN 关键字：需要返回某一个数据值是否位于两个给定的值之间，可以使用范围条件进行检索。通常使用 BETWEEN…AND 和 NOT…BETWEEN…AND 来指定范围条件。

使用 BETWEEN…AND 查询条件时，指定的第一个值必须小于第二个值。因为 BETWEEN…AND 实质是查询条件“大于或等于第一个值，并且小于或等于第二个值”的简写形式。即 BETWEEN…AND 包括两端的值，等价于比较运算符（>=…<=）。

【例 8.16】 在“学生表”中，使用“BETWEEN…AND”关键字查询“总分数”在 85 与 90 之间的学生信息（**实例位置：光盘\TM\sl\8\16**）。

```
select * from 学生表 where 总分数 between 85 and 90;
```

本例运行结果如图 8.21 所示。

学号	姓名	专业	性别	出生日期	总分数	照片	联系方式	备注
14001	马芳	计算机应用技术 ...	女	1989-8-16	89		XML ...	
14005	董敏	计算机应用技术 ...	女	1992-8-5	88		XML ...	

图 8.21　使用 BETWEEN…AND 关键字

而 NOT…BETWEEN…AND 语句返回在两个指定值的范围以外的某个数据值，但并不包括两个指定的值。

【例 8.17】 在“学生表”中，使用“NOT…BETWEEN…AND”关键字查询“总分数”不在 85 与 90 之间的学生信息（**实例位置：光盘\TM\sl\8\17**）。

```
select * from 学生表 where 总分数 not between 85 and 90;
```

本例运行结果如图 8.22 所示。

学号	姓名	专业	性别	出生日期	总分数	照片	联系方式	备注
14002	宋静	计算机应用技术 ...	女	1990-5-12	83		XML ...	
14003	周天星	计算机应用技术 ...	男	1989-2-17	92		XML ...	三好学生，已提...
14004	蒋阳泉	计算机应用技术 ...	男	1991-7-13	84		XML ...	
14006	万方	计算机应用技术 ...	女	1993-8-12	92		XML ...	

图 8.22 使用 NOT…BETWEEN…AND 关键字

（4） IS NULL 关键字：空值（NULL）从技术上来说就是未知的、不确定的值，但空值与空字符串不同，因为空值是不存在的值，而空字符串是长度为 0 的字符串。

因为空值代表的是未知的值，所以并不是所有的空值都相等。例如，学生表中部分学生的备注信息未知，但是无法证明这部分学生备注信息相同。这样就不能用“=”运算符来检测空值。所以 SQL 引入了一个 IS NULL 关键字来检测特殊值之间的等价性，并且 IS NULL 关键字通常在 WHERE 子句中使用。

【例 8.18】 使用 IS NULL 关键字过滤出“学生表”中备注信息为空值的学生信息（**实例位置：光盘\TM\sl\8\18**）。

```
select * from 学生表 where 备注 is null;
```

本例运行结果如图 8.23 所示。

学号	姓名	专业	性别	出生日期	总分数	照片	联系方式	备注
14001	马芳	计算机应用技术 ...	女	1989-8-16	89		XML ...	
14002	宋静	计算机应用技术 ...	女	1990-5-12	83		XML ...	
14004	蒋阳泉	计算机应用技术 ...	男	1991-7-13	84		XML ...	
14005	董敏	计算机应用技术 ...	女	1992-8-5	88		XML ...	
14006	万方	计算机应用技术 ...	女	1993-8-12	92		XML ...	

图 8.23 使用 IS NULL 关键字查询

而 IS NOT NULL 返回的是非空值的数据。

【例 8.19】 使用 IS NOT NULL 关键字过滤出“学生表”中备注信息不为空值的学生信息（**实例位置：光盘\TM\sl\8\19**）。

```
select * from 学生表 where 备注 is not null;
```

本例运行结果如图 8.24 所示。

学号	姓名	专业	性别	出生日期	总分数	照片	联系方式	备注
14003	周天星	计算机应用技术 ...	男	1989-2-17	92		XML ...	三好学生，已提...

图 8.24　使用 IS NOT NULL 关键字

注意

NULL 和 0 或空字符串（''）不是一回事。NULL 是一个特殊的标识符，它用来代表没有信息。而 0 或空字符串（''）是实实在在的值，可以被放在列中，代表一个具体的值。

3. 逻辑筛选

逻辑筛选是指在 WHERE 子句中使用逻辑运算符 AND、OR 和 NOT 进行数据筛选操作，那么这些逻辑运算符可以把多个筛选条件组合起来，这样便于用户获取更加准确的数据记录。

AND 逻辑运算符表示两个逻辑表达式之间“逻辑与”的关系，完全可以使用 AND 运算符加“比较运算符”来代替“BETWEEN…AND”关键字。

【例 8.20】　在“学生表”中使用“AND”运算符查询总分数在 80 与 90 之间的学生信息（**实例位置：光盘\TM\sl\8\20**）。

```
select * from 学生表 where 总分数>=80 and 总分数<=90;
```

本例运行结果如图 8.25 所示。

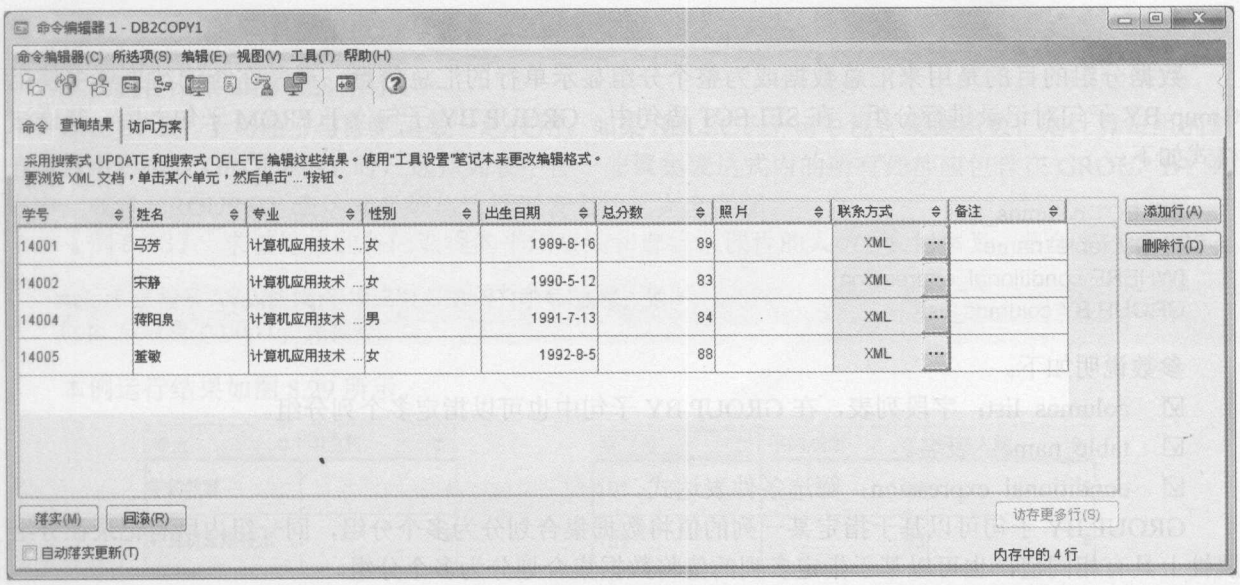


图 8.25　使用 AND 关键字

OR 逻辑运算符表示两个逻辑表达式之间“逻辑或”的关系，两个表达式的结果中有一个为 true，则这个逻辑或表达式的值就为 true。

【例 8.21】　在“学生表”中，使用 OR 逻辑运算符查询总分数小于 85 或大于 90 的学生信息（实

例位置：光盘\TM\sl\8\21）。

```
select * from 学生表 where 总分数<85 or 总分数>90;
```

本例运行结果如图 8.26 所示。

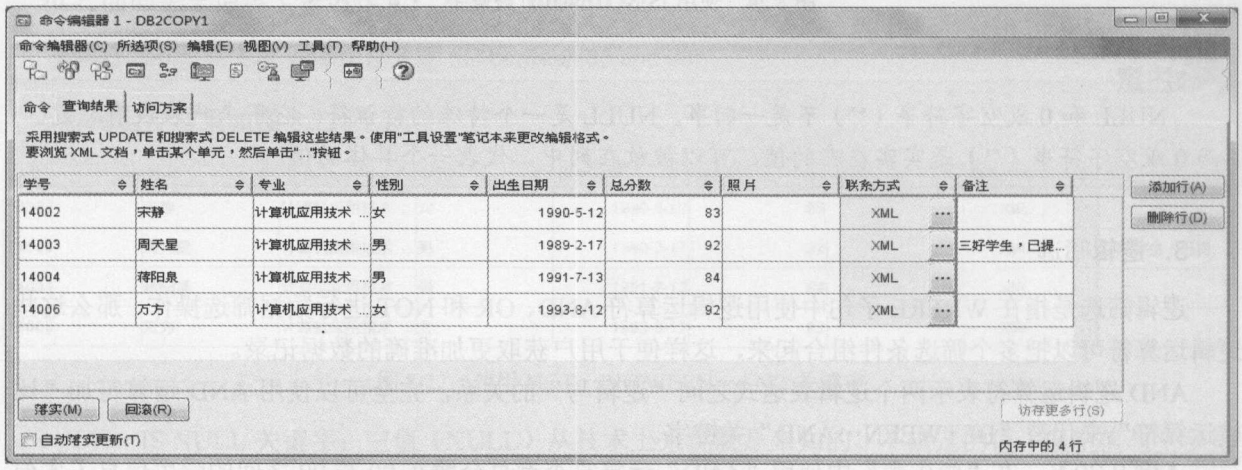


图 8.26　使用 OR 关键字

NOT 逻辑运算符对表达式执行“逻辑非”运算，在前面的实例中已经多次用到过，这里不再赘述。

8.2.3　分组查询

数据分组的目的是用来汇总数据或为整个分组显示单行的汇总信息，通常在查询结果集中使用 Group BY 子句对记录进行分组。在 SELECT 语句中，GROUP BY 子句位于 FROM 子句之后，其语法格式如下：

```
SELECT columns_list
FROM table_name
[WHERE conditional_expression]
GROUP BY columns_list
```

参数说明如下。

☑ columns_list：字段列表，在 GROUP BY 子句中也可以指定多个列分组。

☑ table_name：表名。

☑ conditional_expression：筛选条件表达式。

GROUP BY 子句可以基于指定某一列的值将数据集合划分为多个分组，同一组内所有记录在分组属性上具有相同值，也可以基于指定多列的值将数据集合划分为多个分组。

说明

为了更好地讲解分组查询，编者在数据表中插入了若干条数据，读者朋友可以自己插入数据。

“学生表”添加后数据如图 8.27 所示。



图 8.27　学生表

1. 简单的 GROUP BY 子句

【例 8.22】　求 MRKJ 数据库中各专业的学生数（**实例位置：光盘\TM\sl\8\22**）。

```
select 专业,count(*)学生数 from 学生表 group by 专业;
```

本例运行结果如图 8.28 所示。

GROUP BY 子句经常与聚集函数一起使用。如果 SELECT 子句中包含聚集函数，则计算每组的总值，当用户指定 GROUP BY 时，选择列表中任一非聚集表达式内的所有列都应包含在 GROUP BY 列表中，或者 GROUP BY 表达式必须与选择列表表达式完全匹配。

【例 8.23】　求被选修的各门课程的平均成绩和选修该课程的人数（**实例位置：光盘\TM\sl\8\23**）。

```
select 课程号,AVG(成绩)平均成绩,COUNT(学号)选修人数
from 成绩表 group by 课程号;
```

本例运行结果如图 8.29 所示。

专业	学生数
学前教育	4
计算机应用技术	6

图 8.28　各专业的学生数

课程号	平均成绩	选修人数
101	88	6
102	63	4

图 8.29　显示平均成绩和选修人数

使用 GROUP BY 子句和聚集函数对数据进行分组后，还可以使用 HAVING 子句对分组数据进行进一步筛选。HAVING 子句的格式为：

```
having-clause::=HAVING <search_condition>
```

说明

（1）与 WHERE 子句一样，HAVING 子句一般用来告诉 DB2 数据库管理器如何在一个或多个表中筛选特定的数据。

（2）search_condition 为查询条件。

（3）HAVING 子句与 WHERE 子句的语法类似，但 HAVING 子句可以使用聚合函数。

（4）在 SELECT 语句中，当 WHERE、GROUP BY 与 HAVING 子句都被使用时，要注意它们的作用和执行顺序：WHERE 用于筛选由 FORM 指定的数据对象；GROUP BY 用于对 WHERE 的结果进行分组；HAVING 则是对 GROUP BY 以后的分组数据进行过滤。

【例 8.24】 先求被选修的各门课程的平均成绩和选修该课程的人数，然后通过 HAVING 过滤出平均成绩大于 80 的记录信息（**实例位置：光盘\TM\sl\8\24**）。

```
select 课程号,AVG(成绩)平均成绩,COUNT(学号)选修人数
from 成绩表 group by 课程号 having AVG(成绩)>80;
```

本例运行结果如图 8.30 所示。

【例 8.25】 查找“学前教育”专业平均成绩在 62 以上的学生的学号和平均成绩（**实例位置：光盘\TM\sl\8\25**）。

```
select 学号 ,avg(成绩) 平均成绩 from 成绩表
 where 学号 in (select 学号 from 学生表 where 专业 ='学前教育')
 group by 学号
 HAVING avg(成绩)>=62;
```

本例运行结果如图 8.31 所示。

课程号	平均成绩	选修人数
101	88	6

图 8.30 平均成绩大于 80 的记录信息

学号	平均成绩
14007	64
14008	68
14009	62

图 8.31 实例运行结果

说明

例 8.25 的查询语句分 3 步执行：

（1）首先执行 WHERE 查询条件中的子查询（select 学号 from 学生表 where 专业 ='学前教育'），得到学前教育专业所有学生的学号集合。

（2）然后对“成绩表”中的每条记录，判断其“学号”字段值是否在前面所求得的“学号”集合中。若不在，则跳过该记录，继续处理下一条记录；若在，则加入 WHERE 的结果集。

（3）对“成绩表”筛选完后，按“学号”进行分组，再在各分组记录中选出“平均成绩大于等于 62”的记录，从而形成最后的结果集。

2．带 WITH ROLLUP 关键字的 GROUP BY 子句

使用带 ROLLUP 操作符的 GROUP BY 子句：指定在结果集内不仅包含由 GROUP BY 提供的正常行，还包含汇总行。

【例 8.26】 在 MRKJ 数据库上统计每个专业的男生、女生人数、总人数以及学生总数（**实例位置：光盘\TM\sl\8\26**）。

```
select 专业,性别,COUNT(*)人数
from 学生表 group by 专业,性别 with rollup;
```

本例运行结果如图 8.32 所示。

专业	性别	人数
		10
学前教育		4
计算机应用技术		6
学前教育	女	2
学前教育	男	2
计算机应用技术	女	4
计算机应用技术	男	2

图 8.32　WITH ROLLUP 关键字

说明

结果集中的前 3 行是汇总行，第 1 行是汇总所有专业的总人数，第 2 行是学前教育专业的总人数，第 3 行是计算机应用技术专业的总人数，其他的均为不带 ROLLUP 的 GROUP BY 所产生的行。

从以上含有 ROLLUP 的 GROUP BY 子句的 SELECT 语句的执行结果可以看出，使用 ROLLUP 操作符后，将对 GROUP BY 子句中所指定的各列产生汇总行，产生的规则是：按 GROUP BY 子句中由所有列组成的集合汇总，然后按 GROUP BY 子句中前 2 列汇总，再然后按照 GROUP BY 子句中前 3 列汇总……依此类推。例如本例先根据“专业”和“性别”对“学生表”分组，使用 ROLLUP 后，先对“专业”字段产生了汇总行，然后对 GROUP 子句中的前 2 列“专业”与“性别”组合的不同值产生了汇总行。所产生的汇总行中对应具有不同列值的字段值将置为 NULL。带 ROLLUP 的 GROUP BY 子句可以与复杂的查询条件以及连接查询一起使用。

3．带 CUBE 关键字的 GROUP BY 子句

CUBE 操作符对 GROUP BY 子句中的各列的所有可能组合均产生汇总行。

【例 8.27】 在“学生表”中统计每个专业的男生、女生人数、总人数及男生总数、女生总数、学生总人数（**实例位置：光盘\TM\sl\8\27**）。

```
SELECT 专业,性别,COUNT(*) 人数 from 学生表
GROUP BY 专业,性别 WITH CUBE;
```

本例运行结果如图 8.33 所示。

专业	性别	人数
	女	6
	男	4
		10
学前教育		4
计算机应用技术		6
学前教育	女	2
学前教育	男	2
计算机应用技术	女	4
计算机应用技术	男	2

图 8.33　CUBE 关键字

说明

本例中用于分组的列（也就是 GROUP BY 子句中的列）为专业名和性别，表中专业名有两个值分别为计算机应用技术和学前教育，性别也有两个男和女，再加上 NULL 值，因此它们可能的组合有 5 种，生成 5 个汇总行。

8.2.4　排序查询

在检索数据时，如果把数据从数据库中直接读取出来，这时查询结果将按照默认顺序排列，但往往这种默认排列顺序并不是用户所需要的。尤其返回的数据量较大时，用户查看自己想要的信息非常不方便，因此需要对检索的结果集进行排序。在 SELECT 语句中，可以使用 ORDER BY 子句对检索的结果集进行排序，该子句位于 FROM 子句之后，其语法格式如下：

```
SELECT columns_list
FROM table_name
[WHERE conditional_expression]
[GROUP BY columns_list]
ORDER BY {order_by_expression [ASC | DESC]}
```

参数说明如下。

- ☑ columns_list：字段列表，这里的 GROUP BY 子句中也可以指定多个列分组。
- ☑ table_name：表名。
- ☑ conditional_expression：筛选条件表达式。
- ☑ order_by_expression：表示要排序的列名或表达式。关键字 ASC 表示升序排列，这也是默认

的排序方式；关键字 DESC 表示降序排列。

ORDER BY 子句可以根据查询结果中的一个列或多个列对查询结果进行排序，并且第一个排序项是主要的排序依据，其次那些是次要的排序依据。

【例 8.28】　检索“学生表”中的数据，并按照总分数升序排序（**实例位置：光盘\TM\sl\8\28**）。

```
select 学号,姓名,专业,性别,总分数 from 学生表 order by 总分数;
```

本例运行结果如图 8.34 所示。

学号	姓名	专业	性别	总分数
14010	禁传奇	学前教育	男	60
14009	董湘芬	学前教育	女	62
14007	秦冬雪	学前教育	女	64
14008	欧阳云	学前教育	男	68
14002	宋静	计算机应用技术	女	83
14004	蒋阳泉	计算机应用技术	男	84
14005	董敏	计算机应用技术	女	88
14001	马芳	计算机应用技术	女	89
14003	周天星	计算机应用技术	男	92
14006	万方	计算机应用技术	女	92

图 8.34　升序排列

【例 8.29】　检索“学生表”中的数据，并按照总分数降序排序（**实例位置：光盘\TM\sl\8\29**）。

```
select 学号,姓名,专业,性别,总分数 from 学生表 order by 总分数 desc;
```

本例运行结果如图 8.35 所示。

学号	姓名	专业	性别	总分数
14003	周天星	计算机应用技术	男	92
14006	万方	计算机应用技术	女	92
14001	马芳	计算机应用技术	女	89
14005	董敏	计算机应用技术	女	88
14004	蒋阳泉	计算机应用技术	男	84
14002	宋静	计算机应用技术	女	83
14008	欧阳云	学前教育	男	68
14007	秦冬雪	学前教育	女	64
14009	董湘芬	学前教育	女	62
14010	禁传奇	学前教育	男	60

图 8.35　降序排列

8.2.5 多表关联查询

在实际的应用系统开发中会涉及多个数据表，每个表的信息不是独立存在的，而是若干个表之间的信息存在一定的关系，这样当用户查询某一个表的信息时，很可能需要查询关联数据表的信息，这就是多表关联查询。SELECT 语句自身是支持多表关联查询的，多表关联查询要比单表查询复杂得多。在进行多表关联查询时，可能会涉及表别名、内连接、外连接、自然连接、自连接和交叉连接等概念，下面将对这些内容进行讲解。

回顾下如何创建表，接下来我们要创建部门表（dept）和员工表（emp）。表结构信息如下。

dept 表：

```
create table dept (
deptno character(4) not null,          --部门编号
dname character(20) not null,          --部门名字
location character(20) not null        --部门所在位置
);
```

emp 表：

```
create table emp(
empno character(4) not null,           --编码
ename character(20) not null,          --姓名
job character(20) not null,            --职位
salary smallint not null,              --薪水
bonus smallint ,                       --奖金
hiredate date,                         --入职时间
mgr char(4) ,                          --经理
deptno char(4) not null                --所在部门
);
```

创建完成的 dept 表和 emp 表分别如图 8.36 和图 8.37 所示。

DEPTNO	DNAME	LOCATION
10	研发部	北京
20	财务部	上海
30	销售部	广州
40	后勤部	天津

图 8.36 dept 表

1. 表别名

在多表关联查询时，如果多个表之间存在同名的列，则必须使用表名来限定列的引用。例如，在 dept 表和 emp 表都有 DEPTNO 列，那么当用户使用该列关联查询两个表时，就需要通过制定表名来区

分这两个列的归属。但是，随着查询变得越来越复杂，语句就会因为每次限定列必须输入表名而变得冗长乏味。对于这种情况，SQL 语言提供了设定表别名的机制，使用简短的表别名就可以替代原有较长的表名称，这样就可以大大缩减语句的长度。

EMPNO	ENAME	JOB	SALARY	BONUS	HIREDATE	MGR	DEPTNO
1001	马晓天	Manager	10000	2000	2010-4-10	1005	10
1002	刘丽仁	Analyst	8000	1000	2011-8-11	1001	10
1003	李俊鹏	Analyst	9000	1000	2011-7-10	1001	10
1004	郭得池	Programmer	5000		2011-8-13	1001	10
1005	张东阳	President	15000		2010-5-15		20
1006	陆小川	Manager	5000	400	2009-5-1	1005	20
1007	解灵韵	Clerk	4000	500	2009-5-1	1006	20
1008	郭珊珊	Manager	5000	800	2009-3-12	1005	30
1009	韦宗仁	Salesman	4000		2011-5-20	1008	30
1010	周晓晓	Salesman	4500		2009-5-10	1008	30

图 8.37　emp 表

【例 8.30】　通过 DEPTNO 列来关联 emp 表和 dept 表，并检索这两个表中相关字段的信息（**实例位置：光盘\TM\sl\8\30**）。

```
select e.empno as 员工编号, e.ename as 员工姓名,d.dname as 部门 from emp e,dept d
where e.deptno =d.deptno and e.job ='Manager';
```

本例运行结果如图 8.38 所示。

在上面的 SELECT 语句中，FROM 语句最先执行，然后才是 WHERE 子句和 SELECT 子句，这样当在 FROM 子句中指定表的别名后，当需要限定引用列时，其他所有子句都可以使用表的别名。

另外，还需要注意一点，一旦在 FROM 子句中为表指定了别名，则必须在剩余的子句中都使用表别名，而不允许再使用原来的表名。

2．内连接

内连接是一种常用的多表关联查询方式，一般使用关键字 INNER JOIN 来实现。其中，INNER 关键字可以省略，当只使用 JOIN 关键字时，语句只表示内连接操作。在使用内连接查询多个表时，必须在 FROM 子句之后定义一个 ON 子句，该子句用来指定两个表实现内连接的“连接条件”。需要注意的是，在内连接的检索结果中，所有记录行都是满足连接条件的。内连接语法格式如下：

```
SELECT columns_list
FROM table_name1 [INNER] JOIN table_name2
ON join_condition;
```

参数说明如下。

☑　columns_list：字段列表。

☑ table_name1 和 table_name2：两个要实现内连接的表。

☑ join_condition：实现内连接的条件表达式。

【例 8.31】 通过 DEPTNO 列来内连接 emp 表和 dept 表，并检索这两个表中相关字段的信息（**实例位置：光盘\TM\sl\8\31**）。

```
select e.empno as 员工编号,e.ename as 员工名称,d.dname as 部门
from emp e inner join dept d
on e.deptno = d.deptno;
```

本例运行结果如图 8.39 所示。

员工编号	员工姓名	部门
1001	马晓天	研发部
1006	陆小川	财务部
1008	郭珊珊	销售部

图 8.38 使用 deptno 列关联 emp 和 dept 表

员工编号	员工名称	部门
1001	马晓天	研发部
1002	刘丽仁	研发部
1003	李俊鹏	研发部
1004	郭得池	研发部
1005	张东阳	财务部
1006	陆小川	财务部
1007	解灵韵	财务部
1008	郭珊珊	销售部
1009	韦宗仁	销售部
1010	周晓晓	销售部

图 8.39 内连接

由于上面代码表示内连接操作，所以在 FROM 子句中完全可以省略 INNER 关键字。

3．外连接

使用内连接进行多表查询时，返回的查询结果中只包含符合查询条件和连接条件的行。内连接消除了与另一个表中的任何不匹配的行，而外连接扩展了内连接的结果集，除了返回所有匹配的行外，还会返回一部分或全部不匹配的行，这主要取决于外连接的种类。外连接通常有以下 3 种。

☑ 左外连接：关键字为 LEFT OUTER JOIN 或 LEFT JOIN。

☑ 右外连接：关键字为 RIGHT OUTER JOIN 或 RIGHT JOIN。

☑ 完全外连接：关键字为 FULL OUTER JOIN 或 FULL JOIN。

与内连接不同的是，外连接不只列出与连接条件匹配的行，还能够列出左表（左外连接时）、右表（右外连接时）或两个表（完全外连接时）中所有符合搜索条件的数据行。

左外连接：左外连接的结果集包括 LEFT OUTER 子句中指定的左表的所有行，而不仅仅是连接列所匹配的行。如果左表的某行在右表中没有匹配行，则在相关联的结果行中右表的所有选择列表处置空值。因此，左外连接就是在内连接的基础上加上左表失配的行。

【例 8.32】 在 emp 表和 dept 表之间通过 deptno 列进行左外连接（**实例位置：光盘\TM\sl\8\32**）。

```
select e.empno,e.ename,e.job,d.deptno,d.dname
from emp e left join dept d
on e.deptno=d.deptno;
```

本例运行结果如图 8.40 所示。

EMPNO	ENAME	JOB	DEPTNO	DNAME
1001	马晓天	Manager	10	研发部
1002	刘丽仁	Analyst	10	研发部
1003	李俊鹏	Analyst	10	研发部
1004	郭得池	Programmer	10	研发部
1005	张东阳	President	20	财务部
1006	陆小川	Manager	20	财务部
1007	解灵韵	Clerk	20	财务部
1008	郭珊珊	Manager	30	销售部
1009	韦宗仁	Salesman	30	销售部
1010	周晓晓	Salesman	30	销售部

图 8.40　左外连接

左外连接的查询结果会包含左表中不满足“连接条件”的数据行。

右外连接：右外连接是左外连接的反向连接。将返回右表的所有行。如果右表的某行在左表中没有匹配行，则将为左表返回空值。

右连接与左连接的差别在于：不管左侧表里有没有匹配的记录，它都从右侧表中返回所有记录。因此，右外连接就是在内连接的基础上加上右表失配的行。

【例 8.33】　在 emp 表和 dept 表之间通过 deptno 列进行右外连接（**实例位置：光盘\TM\sl\8\33**）。

```
select e.empno,e.ename,e.job,d.deptno,d.dname
from emp e right join dept d
on e.deptno=d.deptno;
```

本例运行结果如图 8.41 所示。

从上面的查询结果中可以看到，虽然部门编号为 40 的部门在 emp 表中没有员工记录，但它却出现在查询结果中，这说明右外连接的查询结果会包含右表中不满足“连接条件”的数据行。

完全外连接：完整外部连接返回左表和右表中的所有行，全外连接定义为左外连接和右外连接的并集。换句话说，两个表中的所有行都显示在结果集中。如果某个表中没有响应的行，那么该列将返回空值。因此，全外连接就是内连接的基础上加上左表匹配的行和右表失配的行。

【例 8.34】　在 emp 表和 dept 表之间通过 deptno 列进行完全外连接（**实例位置：光盘\TM\sl\8\34**）。

```
select e.empno,e.ename,e.job,d.deptno,d.dname
from emp e full join dept d
on e.deptno =d.deptno;
```

本例运行结果如图 8.42 所示。

EMPNO	ENAME	JOB	DEPTNO	DNAME
1001	马晓天	Manager	10	研发部
1002	刘丽仁	Analyst	10	研发部
1003	李俊鹏	Analyst	10	研发部
1004	郭得池	Programmer	10	研发部
1005	张东阳	President	20	财务部
1006	陆小川	Manager	20	财务部
1007	解灵韵	Clerk	20	财务部
1008	郭珊珊	Manager	30	销售部
1009	韦宗仁	Salesman	30	销售部
1010	周晓晓	Salesman	30	销售部
			40	后勤部

图 8.41　右外连接

4．自连接

在应用系统开发中，用户可能会拥有“自引用式”的外键。“自引用式”外键是指表中的一个列可以是该表主键的一个外键。例如，emp 表中某一行的 mgr 列值（管理者列）可能是另一行的 empno 列值（员工列），因为管理者本身也是公司的员工。这样用户就可以通过 mgr 列与 empno 列的关系，实现查询某个管理者所管理的下属员工信息。

EMPNO	ENAME	JOB	DEPTNO	DNAME
1001	马晓天	Manager	10	研发部
1002	刘丽仁	Analyst	10	研发部
1003	李俊鹏	Analyst	10	研发部
1004	郭得池	Programmer	10	研发部
1005	张东阳	President	20	财务部
1006	陆小川	Manager	20	财务部
1007	解灵韵	Clerk	20	财务部
1008	郭珊珊	Manager	30	销售部
1009	韦宗仁	Salesman	30	销售部
1010	周晓晓	Salesman	30	销售部
			40	后勤部

图 8.42　完全连接

【例 8.35】　查询所有管理者和其所有下属员工的信息（**实例位置：光盘\TM\sl\8\35**）。

```
select em2.ename 上层管理者,em1.ename as 下属员工
from emp em1 left join emp em2
on em1.mgr=em2.empno
order by em1.mgr
```

本例运行结果如图 8.43 所示。

上层管理者	下属员工
	张东阳
马晓天	刘丽仁
马晓天	李俊鹏
马晓天	郭得池
张东阳	马晓天
张东阳	陆小川
张东阳	郭珊珊
陆小川	解灵韵
郭珊珊	韦宗仁
郭珊珊	周晓晓

图 8.43　自连接

5．交叉连接

交叉连接实际上就是不需要任何连接条件的连接，它使用 CROSS JOIN 关键字来实现，其语法格式如下：

```
select columns_list
from table_name1 cross join table_name2;
```

参数说明如下。

☑　colums_list：字段列表。

☑　table_name1 和 table_name2：两个实现交叉连接的表。

【例 8.36】　通过交叉连接 dept 表和 emp 表，计算出查询结果的行数（**实例位置：光盘\TM\sl\8\36**）。

```
select count (*) from dept cross join emp;
```

本例运行结果如图 8.44 所示。

1
40

图 8.44　交叉连接

8.2.6 使用聚集函数计算列值

对表数据进行检索时，经常需要对结果进行汇总计算，例如在 MRKJ 数据库求某门功课的总成绩、统计各分数段的人数等。聚集函数用于计算表中的数据，返回单个计算结果。DB2 所提供的聚集函数如表 8.3 所示。

表 8.3 聚集函数表

函 数 名	说 明
SUM	计算表达式中所有值的总和
AVG	计算表达式中所有值的平均值
MAX	判断表达式中的最大值
MIN	判断表达式中的最小值
CHECKSUM	返回在表的行上或在表达式列表上计算的校验值，用于生成哈希索引
COUNT	计算返回的行数
COUNT_BIG	求组中项数，返回 bigint 类型整数
GROUPING	产生一个附加的列
STDDEV	返回给表达式中所有值的统计标准偏差
VAR，VARIANCE	返回给定表达式中所有值的统计方差
CORR，CORRELATION	返回一对数值的关系系数
COVAR，COVARIANCE	返回一对数值的协方差
ARRAY_AGG	将一组元素聚合到一个数组中

聚集函数语法格式为：

```
function_name([ALL | DISTINCT]expression)
```

参数说明如下。

☑ function_name：是聚集函数名。

☑ expression：是常量、列、函数或表达式，其数据类型只能是 int、smallint、bigint、decimal、double、real 等数字类型。

☑ ALL：表示对所有值进行运算，DISTINCT 表示去除重复值，默认为 ALL。

可以用任何列名或数字表达式替换表达式。除了 COUNT(*)外，在计算返回的聚合值时，每个函数都忽略 NULL 值。

下面分别介绍几个常用的聚合函数。

1. SUM 和 AVG

SUM 和 AVG 分别用来求表达式的总和和平均值。

SUM 和 AVG 的语法如下：

```
SUM/AVG([ALL | DISTINCT] expression)
```

参数说明如下。

☑ Expression：是常量、列、函数或表达式，其数据类型只能是 integer、smallint、bigint、decimal、double 和 real 等数字类型。

☑ ALL：表示对所有值进行运算，DISTINCT 表示去除重复值，默认为 ALL。

☑ SUM/AVG：忽略 NULL 值。

【例 8.37】　求“成绩表”中选修 101 课程的学生的平均成绩（**实例位置：光盘\TM\sl\8\37**）。

```
select AVG(成绩) as 计算机应用技术的平均成绩
 from 成绩表 where 课程号='101';
```

本例运行结果如图 8.45 所示。

AS计算机应用技术的平均成绩
88

图 8.45　AVG 函数

使用 DB2 SQL 语句中的函数作为 SELECT 的计算列时，若不为其指定列标题，则系统将用一个序数表示标题，如“1”“2”。

2. MAX 和 MIN

MAX 和 MIN 分别用于计算表达式中所有值的最大值和最小值。

MAX 和 MIN 函数语法格式为：

```
MAX/MIN([ALL | DISTINCT]expression)
```

参数说明如下。

☑ Expression：是常量、列、函数或表达式，可以是数字、字符和时间日期数据类型。

☑ ALL、DISTINCT 的含义及默认值与 SUM/AVG 函数相同。

☑ MAX/MIN 忽略 NULL 值。

【例 8.38】　求选修 101 课程的学生的最高分和最低分（**实例位置：光盘\TM\sl\8\38**）。

```
select max (成绩) as 计算机应用技术的最高分,
min(成绩) as 计算机应用技术的最低分
from 成绩表 where 课程号='101';
```

本例运行结果如图 8.46 所示。

计算机应用技术的最高分	计算机应用技术的最低分
92	83

图 8.46　MAX 和 MIN 函数

3. COUNT

COUNT 用于统计组中满足条件的行数或总行数。

COUNT 函数的语法格式为：

```
COUNT({[ALL | DISTINCT]expression} | *)
```

参数说明如下。

☑ Expression：是常量、列、函数或表达式，可以是任何数据类型。

☑ ALL、DISTINCT 的含义及默认值与 SUM/AVG 函数相同。

☑ 选择*时将统计总行数，COUNT(*)不需要任何参数。

☑ COUNT 忽略 NULL 值。

【例 8.39】 求学生表中的学生总人数（**实例位置：光盘\TM\sl\8\39**）。

```
select count(*) as 学生总人数 from 学生表;
```

本例运行结果如图 8.47 所示。

【例 8.40】 求选修了课程的学生总人数（**实例位置：光盘\TM\sl\8\40**）。

```
select count(DISTINCT 学号) from 成绩表;
```

本例运行结果如图 8.48 所示。

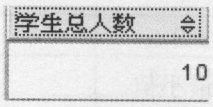


图 8.47 统计学生数

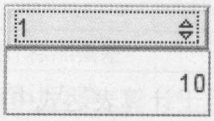


图 8.48 使用 DISTINCT

4. COUNT_BIG

COUNT_BIG 函数的格式、功能与 COUNT 函数都相同，区别仅在于 COUNT_BIG 函数返回 bigint 类型值。

8.3 子查询的用法

在执行数据操作（包括查询、添加、修改和删除等操作）的过程中，如果某个操作需要依赖于另外一个 SELECT 语句的查询结果，那么就可以把 SELECT 语句嵌入到该操作语句中，这样就形成了一个子查询。实际上，在关系数据库中，各表之间的数据关系非常密切，它们相互关联，相互依存，这样就可以根据数据之间的关系使用相应的子查询，从而实现复杂的查询。

8.3.1 子查询

子查询是在 SQL 语句内的另外一条 SELECT 语句，也被称为内查询或是内 SELECT 语句。在 SELECT、INSERT、UPDATE 或 DELETE 命令中允许是一个表达式的地方都可以包含子查询，子查询

甚至可以包含在另外一个子查询中。

【例 8.41】　在 emp 表中查询“部门名称”为“研发部”的员工信息（**实例位置：光盘\TM\sl\8\41**）。

```
select empno,ename,job from emp
where deptno =(select deptno from dept
where dname ='研发部');
```

本例运行结果如图 8.49 所示。

EMPNO	ENAME	JOB
1001	马晓天	Manager
1002	刘丽仁	Analyst
1003	李俊鹏	Analyst
1004	郭得池	Programmer

图 8.49　子查询

对上面的代码进行分析，原本在 emp 表中是不存在 dname（部门名称）字段的，但 emp 表中存在 deptno（部门代码）；dname 字段原本存在于 dept 表中，并且 deptno 字段也存在于 dept 表中，所以 deptno 为两个表之间的关联字段，这样本实例的需求完全可以通过多表关联查询来实现，即可以用下面的代码来代替上面的代码。

```
select empno,ename,job from emp
join dept on emp.deptno =dept.deptno
where dept.dname='研发部'
```

从上面的两段代码中可以看出，相比多表关联查询，子查询的使用更加灵活、功能更强大，而且更容易理解。但是多表关联查询也有它自身的优点，例如，它的查询效率要高于子查询。

在执行子查询操作的语句中，子查询也称为内查询，包含子查询的查询语句也被称为外查询或主查询。在例 8.41 中，下面的语句就是子查询：

```
select deptno from dept where dname ='研发部'
```

外查询就是：

```
select empno,ename,job from emp
```

在一般情况下，外查询语句检索一行，子查询语句需要检索一遍数据，然后判断外查询语句的条件是否满足。如果条件满足，则外查询语句将检索到的数据行添加到结果集中，如果条件不满足，则外查询语句继续检索下一行数据，所以子查询相对多表关联查询要慢一些。

另外，在使用子查询时，还应注意以下规则：

☑　子查询必须用括号“()”括起来。

☑　子查询中不能包括 ORDER BY 子句。

8.3.2 单行子查询

单行子查询是指返回一行数据的子查询语句。当在 WHERE 子句中引用单行子查询时，可以使用单行比较运算符（=、>、<、>=、<=和<>）。

【例 8.42】 在 emp 表中查询既不是最高工资又不是最低工资的员工信息（**实例位置：光盘\TM\sl\8\42**）。

```
select empno,ename,salary from emp
where salary >(select min (salary) from emp )
and salary <(select max(salary) from emp);
```

本例运行结果如图 8.50 所示。

EMPNO	ENAME	SALARY
1001	马晓天	10000
1002	刘丽仁	8000
1003	李俊鹏	9000
1004	郭得池	5000
1006	陆小川	5000
1008	郭珊珊	5000
1010	周晓晓	4500

图 8.50 单行子查询

在上面的语句中，如果内层子查询语句的执行结果为空值，那么外层的 WHERE 子句就始终不会满足条件，这样该查询的结果就必然为空值，因为空值无法参与比较运算。

在执行单行子查询时，要注意子查询的返回结果必须是一行数据，否则 DB2 系统会提示无法执行。另外，子查询中也不能包含 ORDER BY 子句，如果非要对数据进行排序的话，那么只能在外查询语句中使用 ORDER BY 子句。

8.3.3 多行子查询

多行子查询是指返回多行数据的子查询语句。当在 WHERE 子句中使用多行子查询时，必须使用多行运算符（IN、ANY、ALL）。

1. 使用 IN 运算符

当在多行子查询中使用 IN 运算符时，外查询会尝试与子查询结果中的任何一个结果进行匹配，只

要有一个匹配成功，外查询返回当前的搜索记录。

【例 8.43】　在 emp 表中查询不是销售部门的员工信息（**实例位置：光盘\TM\sl\8\43**）。

```
select empno,ename,job from emp
where deptno in
(select deptno from dept where dname <> '销售部');
```

本例运行结果如图 8.51 所示。

EMPNO	ENAME	JOB
1001	马晓天	Manager
1002	刘丽仁	Analyst
1003	李俊鹏	Analyst
1004	郭得池	Programmer
1005	张东阳	President
1006	陆小川	Manager
1007	解灵韵	Clerk

图 8.51　IN 运算符

2. 使用 ANY 运算符

ANY 运算符必须与单行比较运算符结合使用，并且返回行只要匹配子查询的任何一个结果即可。

【例 8.44】　在 emp 表中查询工资大于部门编号为 10 的任意一个员工工资的其他部门的员工信息（**实例位置：光盘\TM\sl\8\44**）。

```
select deptno,ename,salary from emp
where salary > any
(select salary from emp where deptno =10)and deptno <>10;
```

本例运行结果如图 8.52 所示。

DEPTNO	ENAME	SALARY
20	张东阳	15000

图 8.52　ANY 运算符

3. 使用 ALL 运算符

ALL 运算符必须与单行预算符结合使用，并且返回行必须匹配所有子查询的结果。

【例 8.45】　在 emp 表中查询工资大于部门编号为 30 的所有员工工资的员工信息（**实例位置：光盘\TM\sl\8\45**）。

```
select deptno,ename,salary from emp
where salary > all (select salary from emp where deptno =30);
```

本例运行结果如图 8.53 所示。

DEPTNO	ENAME	SALARY
10	马晓天	10000
10	刘丽仁	8000
10	李俊鹏	9000
20	张东阳	15000

图 8.53　ALL 运算符

8.3.4　关联子查询

在单行子查询和多行子查询中，内查询和外查询是分开执行的，也就是说内查询的执行与外查询的执行是没有关系的，外查询仅仅是使用内查询的最终结果。在一些特殊需求的子查询中，内查询的执行需要借助于外查询，而外查询的执行又离不开内查询的执行，这时，内查询和外查询是相互关联的，这种子查询就称为关联查询。

【例 8.46】　在 emp 表中，使用“关联子查询”检索工资大于同职位的平均工资的员工信息（**实例位置：光盘\TM\sl\8\46**）。

```
select empno,ename,salary from emp f
where salary >(select avg (salary) from emp where job=f.job)order by job;
```

本例运行结果如图 8.54 所示。

EMPNO	ENAME	SALARY
1003	李俊鹏	9000
1001	马晓天	10000
1010	周晓晓	4500

图 8.54　关联子查询

在上面的查询语句中，内层查询使用关联子查询计算每个职位的平均工资。而关联子查询必须知道职位的名称，为此外层子查询就使用 f.job 字段值为内层查询提供职位名称，以便计算出某个职位的平均工资。如果外层查询正在检索的数据行的工资高于平均工资，则该行的员工信息会显示出来，否则不显示。

注意

在执行关联子查询的过程中，必须遍历数据表中的每条记录，因此如果被遍历的数据表中有大量数据记录，则关联子查询的执行速度会比较慢。

8.4　小　　结

本章主要讲解了数据库的基本查询，包括简单查询、筛选查询、分组查询、多表联查和子查询等内容。在学习本章内容时，读者应该熟悉选择、投影、连接和集合运算，并熟练掌握检索数据和子查询的用法。

8.5　实践与练习

（1）简述 SELECT 语句的作用。（**答案位置：光盘\TM\sl\8\47**）

（2）WHERE 子句与 HAVING 子句有什么作用？（**答案位置：光盘\TM\sl\8\48**）

（3）DB2 中表的连接查询有哪些种类？（**答案位置：光盘\TM\sl\8\49**）

（4）DB2 中的聚集函数有哪些？（**答案位置：光盘\TM\sl\8\50**）

第 9 章

视图的使用

（视频讲解：37 分钟）

视图是一种常用的数据库对象，它将查询的结果以虚拟表的形式存储在数据库中。视图并不在数据库中以存储数据集的形式存在。视图的结构和内容是建立在对表的查询基础之上的，和表一样包括行和列，这些行列数据都来源于其所引用的表，并且是在引用视图过程中动态生成的。本章主要讲解如何使用“控制中心”和 SQL 语言创建、修改、删除、更新视图以及维护视图。

通过阅读本章，您可以：

- 掌握视图的概念
- 掌握如何创建、修改、查询和更新视图
- 掌握如何维护视图

9.1　视 图 概 述

视图是一个虚拟的表，它由存储的查询构成，可以将它的输出看作一个表。视图同真实表一样，也可以包含一系列带有名称的行和列数据。但是，视图并不在数据库中存储数据值，其数据值来自定义视图的查询语句所引用的表，数据库只在其他视图上，或者同时建立在两者之上。视图看上去非常像数据库中的表，甚至可以在视图中进行 INSERT、UPDATE 和 DELETE 操作。通过视图修改数据时，实际上就是在修改基本表中的数据。与之相对应，改变基本表中的数据也会反映到由该表组成的视图中。

9.1.1　视图与表的关系

视图可以包括它所基于的表中的所有或某些行或列。例如，可以在视图中连接“课程表”和“成绩表”，以便列出特定课程的学生成绩。

视图与表不同，它是一个虚拟表，数据库中存储视图的定义，而不存放视图对应的数据，这些数据仍存放在原来的基本表中。对视图中的数据进行操作时，系统根据视图的定义去操作与视图相关联的基本表，因此，如果基本表中数据发生变化，那么从视图查询出的数据也随之发生变化。从这个意义上说，视图就像一个窗口，透过它可以看到数据库中自己感兴趣的数据及其变化。

当视图列直接派生自基本表的列时，该视图列将继承适用于该表列的所有约束。例如，如果视图包括其基本表的外键，那么该视图的插入和更新操作应遵守与该表相同的引用约束。此外，如果视图的基本表是一个父表，那么该视图的删除和更新操作应遵守与对表执行删除和更新操作相同的规则。

可以从结果表派生出视图中每一列的数据类型，或者从用户定义的结构化类型派生出视图中某列的数据类型，这种视图称为“带类型视图”。某个“带类型视图”（超视图）和从其派生出的所有“带类型视图”（子视图）可以组成视图层次结构。在视图层次结构中，所有“子视图”继承了“超视图”的列。此处的“子视图”指视图层次结构中“带类型视图”（超视图）下的所有“带类型视图”（子视图）中的任何一个。

视图还可用于以下场景：

- ☑ 改变表而不影响应用程序。这可通过创建基于表的视图来完成。使用基础表的应用程序不会因新视图的创建而受到影响。新的应用程序可将创建的视图用于与那些使用基础表的应用程序不同的目的。
- ☑ 对一列中的值求和，选择最大值或计算平均值。
- ☑ 访问一个或多个数据源中的信息。可在 CREATE VIEW 语句内引用昵称，并可创建多个位置/全局视图（全局视图可以连接位于不同系统中多个数据源的信息）。当使用标准的 CREATE VIEW 语法创建引用昵称的视图时，将看到警告，警告目前视图用户的认证标识，而不是视图创建者的认证标识，用于访问数据源处的基本对象。使用 FEDERATED 关键字可以阻止此警告。

9.1.2 视图的优点

视图的优点如下：

- ☑ 为用户集中数据，简化用户的数据查询和处理。有时用户所需要的数据分散在多个数据源中，定义视图可将它们集中在一起，可创建多个位置/全局视图（该视图可以连接位于不同系统中多个数据源的信息），从而方便用户对数据查询和处理。
- ☑ 屏蔽数据库的复杂性。用户不必了解复杂的数据库中的表结构，并且数据表的更改也不影响用户对数据库的使用。
- ☑ 简化用户权限的管理，只需要授予用户使用视图的权限，而不必指定用户只能使用表的特定列，也增加了安全性。例如，可以用视图显示“学生表”的“学号”、“姓名”和“总分数”，而不显示其余的所有列。
- ☑ 便于数据共享。各用户不必都定义和存储自己所需要的数据，可共享数据库的数据，这样同样的数据只需要存储一次。
- ☑ 可以更新数据以便输出到其他应用程序中。

9.1.3 使用视图的注意事项

使用视图时，要注意以下事项：

- ☑ 只有在当前数据库中才能创建视图。
- ☑ 视图的命名必须遵循标识符命名规则，不能和表同名，且对每个用户视图名必须是唯一的，即对不同用户，即使是定义相同的视图，也必须使用不同的名字。
- ☑ 不能把规则、默认值或触发器与视图相关联。
- ☑ 不能在视图上建立任何索引，包括全文索引。

9.2 创建视图

视图可以从一个或多个表、昵称或视图中派生，也可以在检索数据时与表互换使用。在创建视图之前，视图所基于的表、昵称或视图必须已经存在。

9.2.1 使用界面方式创建视图

【例 9.1】 通过“控制中心”创建视图 KCB_VIEW1。

实现步骤如下：

（1）在“控制中心”的对象视图中选择 MRKJ 数据库下的“视图”文件夹，然后单击鼠标右键，

在弹出的快捷菜单中选择“创建”命令，启动“创建视图”对话框，如图 9.1 所示。

（2）在“创建视图”对话框中，选择默认的视图模式为“MR”，输入视图名为“KCB_VIEW1”，如图 9.2 所示。

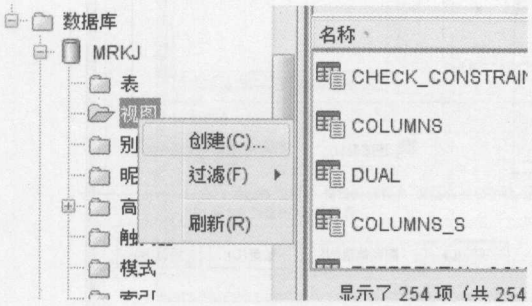


图 9.1　创建视图菜单

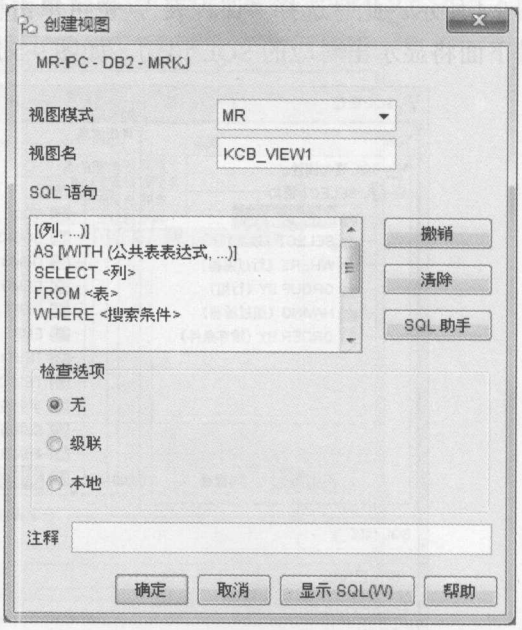


图 9.2　“创建视图”对话框

（3）单击“SQL 助手”按钮，将打开“SQL 语句”对话框，如图 9.3 所示。

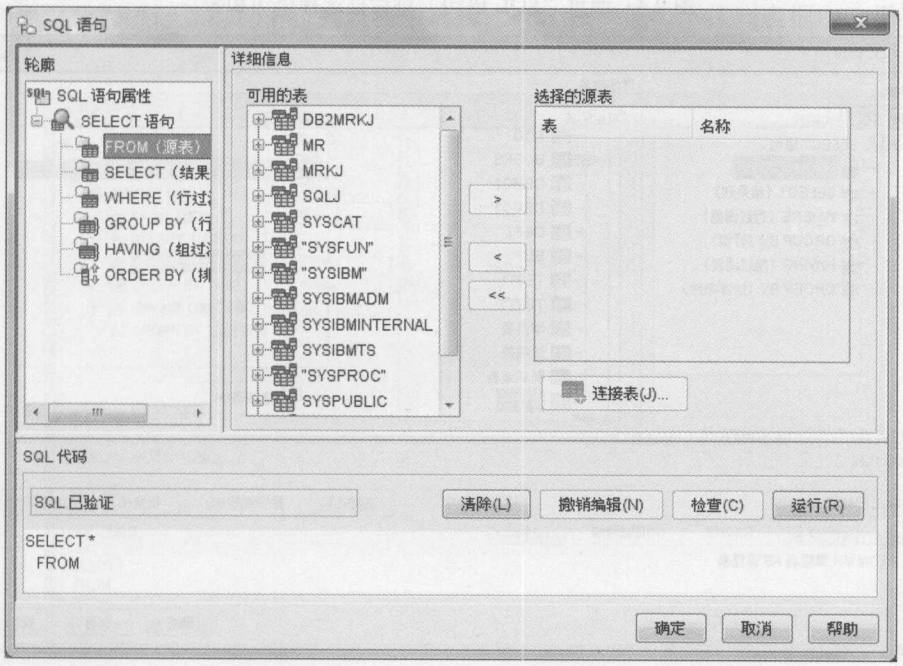


图 9.3　“SQL 语句”对话框

通过选择相应的 SQL 语句属性，将在“详细信息”选项区域显示相对应的信息，选择所需要的信息即可完成 SQL 语句。

（4）选择“FROM（源表）”选项，在“详细信息”选项区域中列出的“可用的表”中选择“课程表”，如图 9.4 所示。此时选择“课程表”，也可单击移动按钮“>”，将“课程表”添加到“选择的源表”中，此时下面将显示出对应的 SQL 语句，如图 9.5 所示。

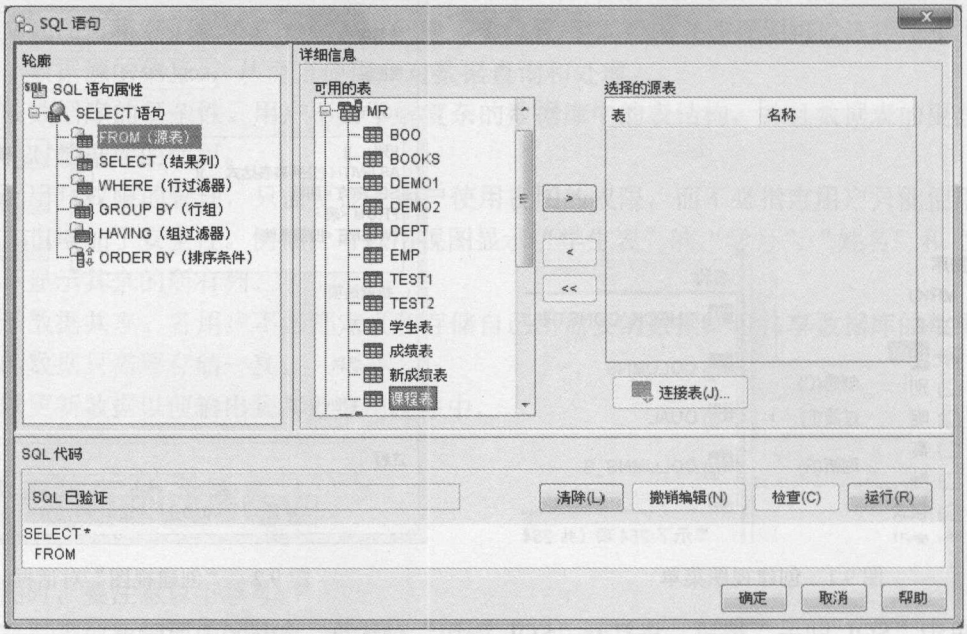


图 9.4　通过“SQL 语句”对话框选择语句和表

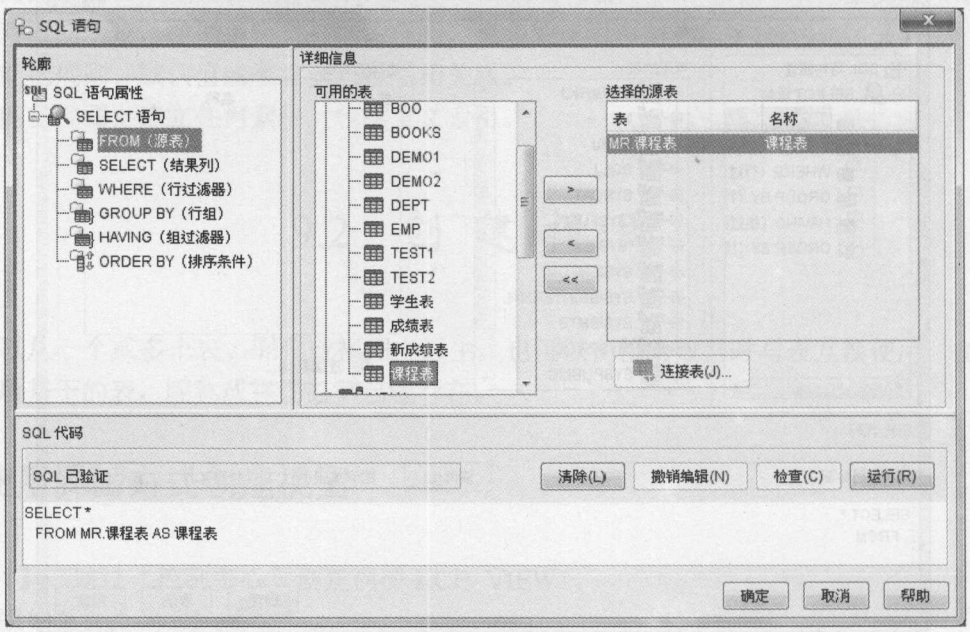


图 9.5　FORM（源表）

（5）选择“SELECT（结果列）”选项，将在“详细信息”选项区域中列出“可用的列”，此时根据需要将“可用的列”列表中的项目添加到“结果列”下面的列表中，如果要选择某个表中某一列，则在“可用的列”列表中选择这个列名，例如“课程号”，然后单击“>”按钮；如果要选择某个表的所有列，则在“可用的列”列表中任意选择表名或列名，然后单击“>>”按钮，如图 9.6 所示。

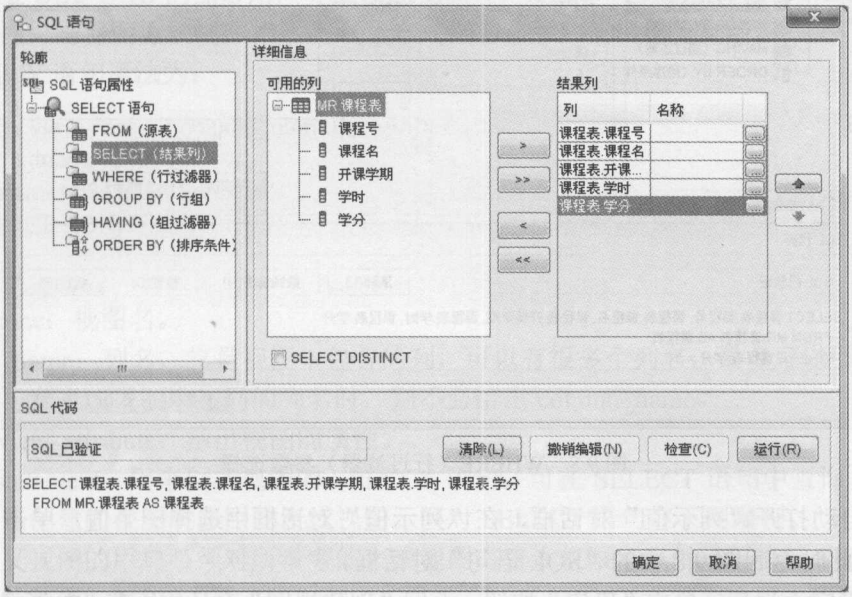


图 9.6　SELECT（结果列）

（6）选择“WHERE（行过滤器）”选项，如图 9.7 所示。在“详细信息”选项区域中的“列”下拉列表框中选择“学分”，“运算符”下拉列表框中选择“=”，在“值”下拉列表框中选择列值，如图 9.8 所示。

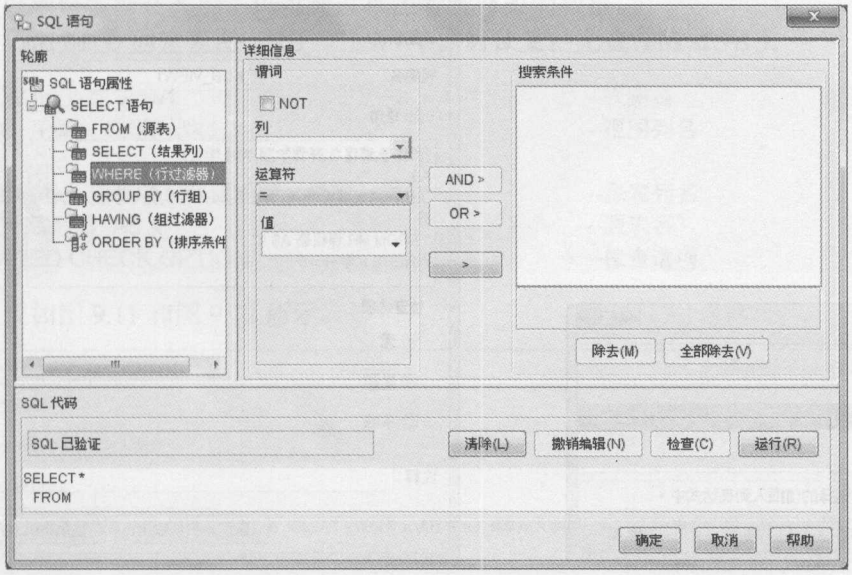


图 9.7　WHERE（行过滤器）

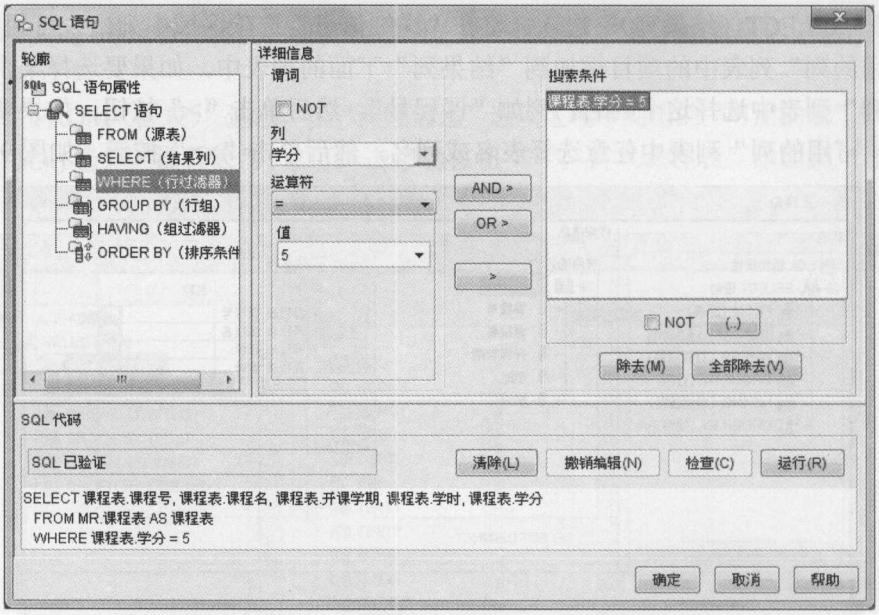


图 9.8　WHERE（行过滤器）参数选择

（7）此时自动打开“列示值”对话框，在“列示值”对话框中选择一个值，单击“确定”按钮，如图 9.9 所示，此时返回图 9.8 中的“SQL 语句”对话框。

（8）检查确定无误后，单击“确定”按钮，返回“创建视图”对话框。在“检查选项”区选中“级联”单选按钮。修改“SQL 语句”代码框中的代码，修改完成后单击“确定”按钮，完成视图“KCB_VIEW1”的创建，如图 9.10 所示。

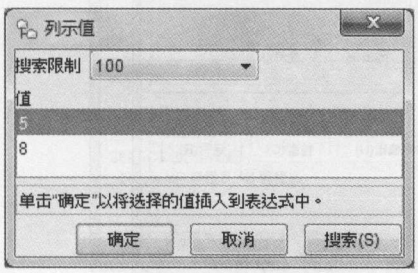


图 9.9　“列示值”对话框

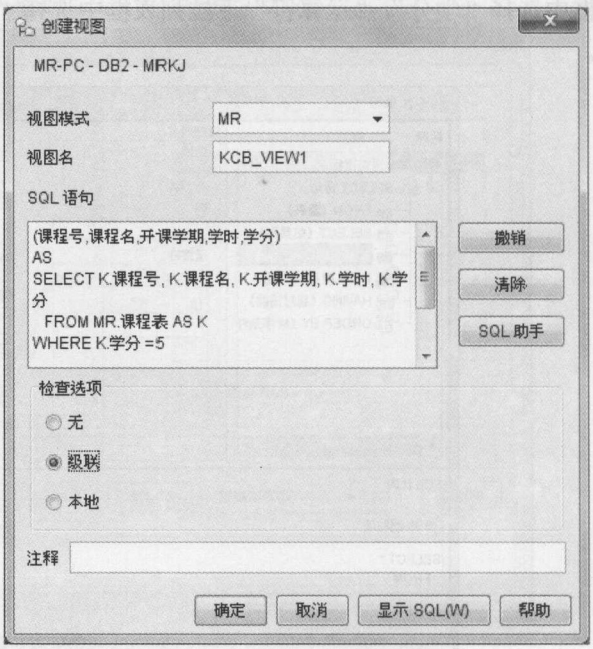


图 9.10　“创建视图”对话框

9.2.2　使用命令方式创建视图

除了可以通过 DB2 UDB 的图形用户界面创建视图外，还可以使用 CREATE VIEW 语句（即命令方式）来创建视图。与界面方式视图相比，命令方式更为常用，使用也更灵活。

CREATE VIEW 语句语法为：

```
CREATE VIEW view_name [({column_name}[,……n])]
 [WITH {view_attribute}[,……n]]
AS select_statement
[WITH CHECK OPTION]
```

参数说明如下。

- ☑ view_name：视图名。
- ☑ column_name：列名，它是视图中包含的列，可以有很多个列名，最多可以引用 1024 个列。当使用与源表或视图中相同的列名时，则不必给出 column_name。
- ☑ WITH view_attribute：指出视图的属性。
- ☑ select_statement：用来创建视图的 SELECT 语句，可在 SELECT 语句中查询多个表或视图，以表明新创建的视图所参照的表或视图。但对 SELECT 语句有下列限制。
  - ➢ 定义视图的用户必须对所参考的表或视图有查询权限，也就是可执行 SELECT 语句。
  - ➢ 不能使用 ORDER BY 子句。
  - ➢ 不能使用 INTO 子句。
  - ➢ 不能在临时表或表变量上创建视图。
- ☑ WITH CHECK OPTION：指出在视图上进行 UPDATE、INSERT 和 DELETE 操作时都要符合 select_statement 所指定的限制条件，即保证修改、插入和删除的行满足视图定义中的条件，这样可以确保数据修改后，仍可通过视图看到修改的数据。

【例 9.2】　通过命令创建视图 KCB_VIEW1（**实例位置：光盘\TM\sl\9\1**）。

```
CREATE VIEW mr.kcb_view1                          --视图名
(课程号,课程名,开课学期,学时,学分)                --视图列名
AS
 SELECT k.课程号,k.课程名,k.开课学期,k.学时,k.学分   --源表列名
     FROM mr.课程表  AS k                          --源表名
WITH CASCADED CHECK OPTION                        --检查选项
```

本例运行结果如图 9.11 和图 9.12 所示。

```
---------- 输入的命令 ----------
CREATE VIEW mr.kcb_view1
(课程号,课程名,开课学期,学时,学分)
AS
SELECT k.课程号,k.课程名,k.开课学期,k.学时,k.学分
FROM 课程表 AS k
WITH CASCADED CHECK OPTION;
--------------------------------------------------------------------------------
CREATE VIEW mr.kcb_view1 (课程号,课程名,开课学期,学时,学分) AS SELECT k.课程号,k.课程名,k.开课学期,k.学时,k.学分 FROM 课程表 AS k WITH CASCADED CHECK OPTION
DB20000I  SQL 命令成功完成。
```

图 9.11　创建视图 mr.kcb_view1

课程号	课程名	开课学期	学时	学分
101	计算机应用技术 ...	5	72	5
102	学前教育 ...	5	64	8

图 9.12　查看视图 mr.kcb_view1

9.2.3　创建特殊类型视图

1. 包含检查选项的视图

定义了 WITH CHECK OPTION 的视图将针对该视图的 SELECT 语句强制检查任何修改或插入的行。使用检查选项的视图也称为“对称视图”。

例如，仅返回课程号 101 中的学生成绩的对称视图不允许插入其他课程成绩。因此，此选项将确保数据库中修改的数据的完整性，并在 INSERT 或 UPDATE 操作期间违反条件时返回错误。

如果应用程序无法将需要的规则定义为表检查约束，或者规则不适用于数据的所有用法，那么可以使用另一种方法在应用程序逻辑中实施规则。可以考虑创建一个表视图，表视图对数据的条件包括在指定的 WHERE 子句和 WITH CHECK OPTION 子句中。此视图定义将数据检索限于对应用程序有效的行集。此外，如果可以更新表视图，那么 WITH CHECK OPTION 子句将更新、插入和删除操作仅限于适用于应用程序的行。

注意

在未指定 WITH CHECK OPTION 的情况下，在修改期间不会进行数据验证。

不能对下列视图指定 WITH CHECK OPTION：

- ☑ 使用只读选项定义的视图（只读视图）。
- ☑ 引用 NODENUMBER 或 PARTITION 函数、非确定性函数（例如 RAND）或使用外部操作的函数的视图。
- ☑ 带类型的视图。

【例 9.3】 创建视图 CJB_VIEW1，该视图包含“计算机基础”课程的学生信息（**实例位置：光盘\TM\sl\9\2**）。

```
CREATE VIEW mr.cjb_view1
AS
SELECT c.学号,c.课程号,c.成绩
FROM mr.成绩表 c
WHERE c.课程号='101'
WITH CHECK OPTION;
```

本例运行结果如图 9.13 所示。

```
---------- 输入的命令 ----------
CREATE VIEW mr.cjb_view1
AS
SELECT c.学号,c.课程号,c.成绩
FROM mr.成绩表 c
WHERE c.课程号='101'
WITH CHECK OPTION;
--------------------------------------------------------------------------------
CREATE VIEW mr.cjb_view1 AS SELECT c.学号,c.课程号,c.成绩 FROM mr.成绩表 c WHERE c.课程号='101' WITH CHECK OPTION
DB20000I  SQL 命令成功完成。
```

图 9.13　创建视图 mr.cjb_view1

说明

以上是一个使用 WITH CHECK OPTION 的视图定义的示例。需要此选项以确保始终检查条件。该视图确保“课程号”始终为“101”，这将限制“课程号”列的输入值。使用视图来插入新值时，始终强制执行“WITH CHECK OPTION”，如果在 INSERT 语句中使用此视图，那么当“课程号”列的值不是“101”时将拒绝执行，如图 9.14 所示。

```
SQL0161N 插入或更新操作的结果行不符合视图定义。  SQLSTATE=44000

SQL0161N 插入或更新操作的结果行不符合视图定义。

说明:

在作为插入或更新操作的对象的视图定义中指定了 WITH CHECK OPTION 子句，或
者 FROM 子句指定了将视图作为目标的插入或更新操作。因此，检查在该视图中
插入或更新行的所有尝试，以确保其结果符合视图定义。

如果 FROM 子句中包含以视图作为目标的插入或更新操作，那么始终将当作视图
已经定义了 WITH CHECK OPTION 一样来处理插入或更新操作。
```

图 9.14　插入或更新的操作不符合视图定义

【例 9.4】　创建学生基本信息视图“xsb_view1”，该视图中包含出生日期小于“1991-7-13”的学生的“学号”“姓名”“专业”“性别”“出生时间”“总分数”信息（**实例位置：光盘\TM\sl\9\3**）。

```
CREATE VIEW mr.xsb_view1
AS
SELECT x.学号,x.姓名,x.专业,x.性别,x.出生日期,x.学分
FROM mr.学生表  as x
WHERE x.出生日期  < '1991-7-13'
WITH CHECK OPTION;
```

本例运行结果如图 9.15 和图 9.16 所示。

说明

该视图确保学生出生日期始终小于'1991-7-13'，这将限制“出生时间”列的输入值，使用视图来插入新值时，始终强制执行“WITH CHECK OPTION”。

如果在 SELECT 语句中使用此视图，那么将会调用条件（WHERE 子句），并且生成的表仅包含匹配的数据行。也就是说“WITH CHECK OPTION”不影响 SELECT 语句的结果。

```
---------- 输入的命令 ----------
CREATE VIEW mr.xsb_view1
AS
SELECT x.学号,x.姓名,x.专业,x.性别,x.出生日期,x.总分数
FROM mr.学生表 as x
WHERE x.出生日期 <'1991-7-13'
WITH CHECK OPTION;
--------------------------------------------------------------------------------
CREATE VIEW mr.xsb_view1 AS SELECT x.学号,x.姓名,x.专业,x.性别,x.出生日期,x.总分数 FROM mr.学生表 as x WHERE x.出生日期 <'1991-7-13' WITH CHECK OPTION
DB20000I  SQL 命令成功完成。
```

图 9.15　创建视图 mr.xsb_view1

学号	姓名	专业	性别	出生日期	总分数
14001	马芳	计算机应用技术 ...	女	1989-8-16	89
14002	宋静	计算机应用技术 ...	女	1990-5-12	83
14003	周天星	计算机应用技术 ...	男	1989-2-17	92
14007	秦冬雪	学前教育 ...	女	1989-5-6	64
14009	董淑芬	学前教育 ...	女	1989-8-16	62
14010	禁传奇	学前教育 ...	男	1989-12-8	60

图 9.16　查看视图 mr.xsb_view1

2. 系统目录视图

数据库管理器维护一组表和视图，这些表和视图包含关于数据库管理器所控制数据的信息。这些表和视图统称为系统目录。系统目录包含关于数据库对象（例如表、视图、索引、程序包和函数）的逻辑和物理结构信息，还包含统计信息。数据库管理器确保系统目录中的描述始终正确。

系统目录视图类似于其他数据库视图。可以使用 SQL 语句来检查系统目录视图中的数据。可以使用一组可更新的系统目录视图来修改系统目录中的某些值。

3. 嵌套视图定义

如果一个视图基于另一个视图（即这两个视图是嵌套的视图，其中前一个视图是子视图，后一个视图是超视图），那么必须求值的谓词数目取决于指定的 WITH CHECK OPTION。

（1）如果未使用 WITH CHECK OPTION 定义视图，那么在检查任何插入或更新操作的数据有效性时不会使用该视图的定义。但是，如果视图直接或间接基于另一个使用 WITH CHECK OPTION 定义的视图，那么在检查任何插入或更新操作时将使用该超视图的定义。

（2）如果视图是使用 WITH CASCADED CHECK OPTION 或仅用 WITH CHECK OPTION（CASCADED 是 WITH CHECK OPTION 的默认值）定义的，那么检查任何插入或更新操作时，将使用该视图的定义。此外，该视图（子视图）将继承它所基于的任何可更新视图（超视图）的搜索条件，即使这些超视图不包含 WITH CHECK OPTION，也继承这些条件。

例如，如果视图 V2 基于视图 V1，并且 V2 的检查选项是使用 WITH CASCADED CHECK OPTION 定义的，那么在对视图 V2 执行 INSERT 和 UPDATE 语句时，要同时满足这两个视图各自的搜索条件（相当于在两个视图的搜索条件之间用了“AND”谓词）。

（3）在创建视图时，还可以指定 WITH LOCAL CHECK OPTION。如果视图是使用 LOCAL CHECK OPTION 定义的，那么在检查任何插入或更新操作时将使用该视图的定义。但是，该视图不继承它所

基于的任何可更新视图的搜索条件。

【例 9.5】　在视图 xsb_view1 的基础上创建视图 xsb_view2，视图 xsb_view2 限定“总学分>50”（实例位置：光盘\TM\sl\9\4）。

```
CREATE VIEW mr.xsb_view2
AS
SELECT 学号,姓名,专业,性别,出生日期,总分数
FROM mr.xsb_view1
WHRERE 总分数 > 50
WITH CASCADED CHECK OPTION;
```

本例运行结果如图 9.17 和图 9.18 所示。

```
---------- 输入的命令 ----------
CREATE VIEW mr.xsb_view2
AS
SELECT 学号,姓名,专业,性别,出生日期,总分数
FROM mr.xsb_view1
WHERE 总分数 > 50
WITH CASCADED CHECK OPTION;
--------------------------------------------------------------------------------
CREATE VIEW mr.xsb_view2 AS SELECT 学号,姓名,专业,性别,出生日期,总分数 FROM mr.xsb_view1 WHERE 总分数 > 50 WITH CASCADED CHECK OPTION
DB20000I  SQL 命令成功完成。
```

图 9.17　创建视图 mr.xsb_view2

学号	姓名	专业	性别	出生日期	总分数
14001	马芳	计算机应用技术	女	1989-8-16	89
14002	宋静	计算机应用技术	女	1990-5-12	83
14003	周天星	计算机应用技术	男	1989-2-17	92
14007	秦冬雪	学前教育	女	1989-5-6	64
14009	董淑芬	学前教育	女	1989-8-16	62
14010	禁传奇	学前教育	男	1989-12-8	60

图 9.18　查看视图 mr.xsb_view2

说明

（1）以上示例显示了使用 WITH CASCADED CHECK OPTION 的 CREATE VIEW 语句。视图 XSB_VIEW2 是在视图 XSB_VIEW1 的基础上创建的，而 XSB_VIEW1 是使用 WITH CHECK OPTION 创建的。如果要在 XSB_VIEW2 中插入或更新记录，那么该记录的值应该限定“出生日期<1991-7-13”和“总分数>50”。

（2）即使视图 XSB_VIEW2 不包含 WITH CHECK OPTION，在对 XSB_VIEW2 执行插入或更新操作时也会强制执行条件“出生日期<1991-7-13”。

4．可删除视图

根据定义视图的方式，视图可以是可删除视图。可删除视图是可以对其成功发出 DELETE 语句的视图。

只有在遵循下列规则的情况下，视图才能被视为可删除视图。

- ☑ 外部查询的每个FROM子句仅标识一个表(不带有OUTER子句)、可删除视图(不带有OUTER子句)、可删除的嵌套表表达式或可删除的公共表表达式。
- ☑ 数据库管理器应该能够使用视图定义来派生表中要删除的行。以下操作使视图变得不可删除：
  - ➢ 使用GROUP BY子句或列函数将多行分组为一行将导致原始行丢失并使得视图不可删除。
  - ➢ 同样，从VALUES派生行时，没有要删除行的表。视图也将不可删除。
- ☑ 外部全查询不适用GROUP BY或HAVING子句。
- ☑ 外部全查询的选择列表中不包括列函数。
- ☑ 外部全查询不使用集操作（UNION、EXCEPT或INTERSECT），但UNION ALL除外。
- ☑ UNION ALL的操作数中的表不能是相同的表，并且每个操作数必须可删除。
- ☑ 外部全查询的选择列表不包括DISTINCT。

视图必须符合上面所述的所有规则才能被视为可删除视图。例如，下列视图是可删除视图，因为其遵循可删除视图的所有规则。

【例 9.6】 创建视图cjb_view2，该视图包含学号为“14002”的学生成绩信息（**实例位置：光盘\TM\sl\9\5**）。

```
CREATE VIEW mr.cjb_view2
AS
SELECT 学号,课程号,成绩
FROM mr.成绩表
WHERE 学号 ='14002';
```

本例运行结果如图9.19和图9.20所示。

```
---------- 输入的命令 ----------
CREATE VIEW mr.cjb_view2
AS
SELECT 学号,课程号,成绩
FROM mr.成绩表
WHERE 学号 ='14002';
--------------------------------------------------------------------------------
CREATE VIEW mr.cjb_view2 AS SELECT 学号,课程号,成绩 FROM mr.成绩表 WHERE 学号 ='14002'
DB20000I SQL 命令成功完成。
```

图 9.19 创建视图 mr.cjb_view2

学号	课程号	成绩
14002	101	83

图 9.20 查看视图 mr.cjb_view2

5. 可插入视图

可插入视图允许使用视图定义来插入行。如果为视图定义了用于插入操作的INSTEAD OF触发器，或者视图中的至少一列更新（与用于更新的INSTEAD OF触发器无关），并且视图的全查询不包括UNION ALL，那么视图是可插入视图。当且仅当给定行正好满足基础表的检查约束时，才能将行插入到视图中（包括UNION ALL）。要插入到包含不可更新列的视图中，必须从列的列表中省略这些列。

【例 9.7】 创建视图cjb_view3，该视图包含“成绩表”中的“学号”和“成绩”列（**实例位置：**

光盘\TM\sl\9\6)。

```
CREATE VIEW mr.cjb_view3
AS
SELECT 学号, 成绩
FROM mr.成绩表;
```

本例运行结果如图 9.21 和图 9.22 所示。

```
---------------------- 输入的命令 ----------------------
CREATE VIEW mr.cjb_view3
AS
SELECT 学号,成绩
FROM mr.成绩表;
----------------------------------------------------------
CREATE VIEW mr.cjb_view3 AS SELECT 学号,成绩 FROM mr.成绩表
DB20000I  SQL 命令成功完成。
```

图 9.21　创建视图 mr.cjb_view3

本例创建的视图是一个可插入视图。但是在尝试插入视图时将会失败，提示信息如图 9.22 所示。这是因为表中不存在接受空值的列。这些列中的某些列未出现在视图定义中，尝试使用视图来插入值时，数据库管理器会尝试将一个空值插入到 NOT NULL 列中，系统不允许执行此操作。

```
SQL0407N  不允许对 NOT NULL 列 "TBSPACEID=4, TABLEID=6, COLNO=1                    " 赋予空值。

说明:

发生了下列情况之一：
* 更新或插入值为 NULL，但是对象列在表定义中已声明为 NOT NULL。因此：
  * 空值不可插入到该列中。
  * 更新不能将该列中的值设置为 NULL。
  * 触发器中的 SET 转换变量语句不能将该列中的值设置为 NULL。

* 更新或者插入值是 DEFAULT，但是对象列被声明为 NOT NULL，而表定义中又
  没有 WITH DEFAULT。因此：
  * 缺省值 NULL 不能插入该列。
  * 更新不能在该列设置缺省值 NULL。
  * 触发器中的 SET 转换变量语句不能在该列中设置缺省值 NULL。

* INSERT 语句的列名列表省略了一列，该列声明为 NOT NULL，而表定义中没有
  WITH DEFAULT。
* INSERT 语句的视图省略了一列，该列声明为 NOT NULL 而基本表定义中没有
  WITH DEFAULT。
* ALTER TABLE 语句试图使用 SET NOT NULL 来改变列，但是现有表数据包含一
  个具有空值的行。
```

图 9.22　不允许将空值插入 NOT NULL 列

注意

对表定义的约束与可以使用基于该表的视图执行的操作无关。

6. 可更新视图

可更新视图是一种特殊的可删除视图。如果可删除视图中的至少一列可更新，那么该可删除视图就变成可更新视图。

当满足下列所有规则时，视图中的一列将可更新：

☑ 视图是可删除视图。

☑ 列解析为表列（不使用解引用操作）并且为指定 READ ONLY 选项。

☑ 如果视图的全查询包含 UNION ALL，那么 UNION ALL 的操作数的所有相应列具有完全匹配的数据类型（包括长度或精度和小数位）以及完全匹配的默认值。

【例 9.8】 创建视图 cjb_view4，该视图包含“成绩表”中的“成绩”为 92 分的所有成绩记录的“学号”、“课程号”和“成绩”列（**实例位置：光盘\TM\sl\9\7**）。

```
CREATE VIEW mr.cjb_view4
AS
SELECT 学号,课程号, 成绩
FROM mr.成绩表
WHERE 成绩=92
WITH CASCADED CHECK OPTION;
```

本例运行结果如图 9.23 和图 9.24 所示。

```
---------- 输入的命令 ----------
CREATE VIEW mr.cjb_view4
AS
SELECT 学号,课程号,成绩
FROM mr.成绩表
WHERE 成绩=92
WITH CASCADED CHECK OPTION;
------------------------------------------------------------------------
CREATE VIEW mr.cjb_view4 AS SELECT 学号,课程号,成绩 FROM mr.成绩表 WHERE 成绩=92 WITH CASCADED CHECK OPTION
DB20000I  SQL 命令成功完成。
```

图 9.23 创建视图 mr.cjb_view4

学号	课程号	成绩
14003	101	92
14006	101	92

图 9.24 查看视图 mr.cjb_view4

说明

本例使用的是无法更新的常量值。但是，视图是可删除视图并且该视图中的至少一列可更新。因此，视图是可更新视图。

7. 只读视图

如果一个视图不可删除、更新或插入，那么该视图是只读视图。如果一个视图不符合删除视图的至少一个规则，那么该视图是只读视图。

SYSCAT.VIEWS 系统目录视图中的 READONLY 列指示视图是只读（R）视图。

【例 9.9】 创建视图 xsb_cjb_read_only_view，该视图包含选修了“101”（课程号）课程的学生的“学号”“姓名”“专业”“课程号”“成绩”信息（**实例位置：光盘\TM\sl\9\8**）。

```
CREATE VIEW mr.xsb_cjb_read_only_view
as
```

```
select distinct x.学号,x.姓名,x.专业,c.课程号,c.成绩
FROM mr.学生表 AS x, mr.成绩表 AS c
WHERE c.课程号='101' AND x.学号=c.学号;
```

本例运行结果如图 9.25 和图 9.26 所示。

```
---------- 输入的命令 ----------
CREATE VIEW mr.xsb_cjb_read_only_view
as
select distinct x.学号,x.姓名,x.专业,c.课程号,c.成绩
FROM mr.学生表 AS x, mr.成绩表 AS c
WHERE c.课程号='101' AND x.学号=c.学号;
--------------------------------------------------------------------------------
CREATE VIEW mr.xsb_cjb_read_only_view as select distinct x.学号,x.姓名,x.专业,c.课程号,c.成绩 FROM mr.学生表 AS x, mr.成绩表 AS c WHERE c.课程号='101' AND x.学号=c.学号
DB20000I  SQL 命令成功完成。
```

图 9.25　创建视图 mr.xsb_cjb_read_only_view

14001	马芳	计算机应用技术 ...	101	89
14002	宋静	计算机应用技术 ...	101	83
14003	周天星	计算机应用技术 ...	101	92
14004	蒋阳泉	计算机应用技术 ...	101	84
14005	董敏	计算机应用技术 ...	101	88
14006	万方	计算机应用技术 ...	101	92

图 9.26　查看视图 mr.xsb_cjb_read_only_view

说明

上面的示例不是可删除视图，因为它使用了 DIASTINCT 子句且 SELECT 语句涉及多个表。

9.3　查 询 视 图

视图定义完成后，就可以和查询基本表那样对视图进行查询。

执行对视图的查询时，首先进行有效的检查，检查要查询的表、视图是否存在。如果存在，那么从系统表中取出视图的定义，把定义中的子查询和用户的查询结合起来，转换成等价的对基本表的查询，然后再执行转换后的查询。

对视图的查询操作与对基本表的查询操作基本一样，可以在“控制中心”通过界面操作方式查询，也可以直接通过“命令编辑器”用命令来查询。

注意

（1）使用视图查询时，若其关联的基本表中添加了新的字段，则必须重新创建视图才能查询到字段。

（2）如果与视图相关联的表或视图被删除，则该视图将不能再使用。

【例 9.10】 在“控制中心”通过界面查询视图。

（1）在“控制中心”对象视图中，选择 MRKJ 数据库下的“视图”文件夹，在内容窗格中查找要查询的视图，例如查询 XSB_VIEW1。单击鼠标右键，在弹出的快捷菜单中选择“查询”命令，如图 9.27 所示。

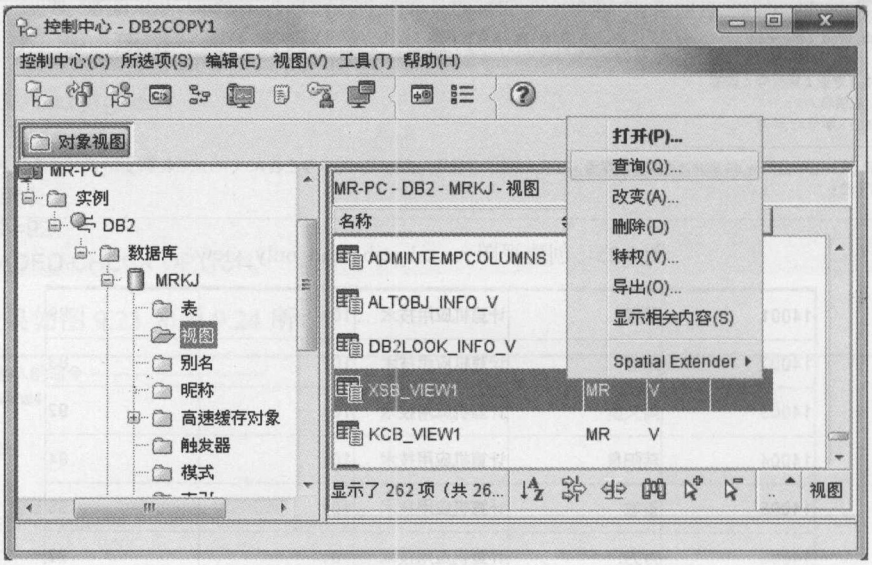


图 9.27 查询视图菜单

（2）在选择“查询”命令后，将打开“命令编辑器”窗口，系统已经将默认的 SQL 语句显示出来，如图 9.28 所示。此时可以根据实际情况修改 SQL 语句来查询视图，最后单击▶按钮将执行查询视图的操作。

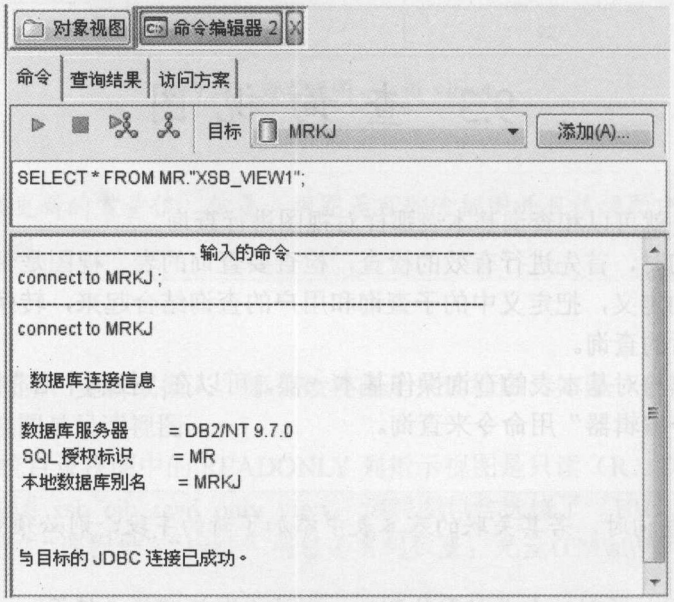


图 9.28 命令编辑器中自动生成的 SQL 查询语句

（3）在系统默认显示出 SQL 查询语句时，单击▶按钮将执行查询 XSB_VIEW1 视图的操作，查询结果如图 9.29 所示。

学号	姓名	专业	性别	出生日期	总分数
14001	马芳	计算机应用技术 ...	女	1989-8-16	89
14002	宋静	计算机应用技术 ...	女	1990-5-12	83
14003	周天星	计算机应用技术 ...	男	1989-2-17	92
14007	秦冬雪	学前教育 ...	女	1989-5-6	64
14009	董淑芬	学前教育 ...	女	1989-8-16	62
14010	禁传奇	学前教育 ...	男	1989-12-8	60

图 9.29　查询 XSB_VIEW1 视图

【例 9.11】　查询“计算机应用技术”专业的女同学的学生信息（**实例位置：光盘\TM\sl\9\9**）。

本例是对视图 XSB_VIEW1 进行查询，在“命令编辑器”中直接输入以下 SQL 命令，然后单击▶按钮执行。

```
SELECT * FROM mr.xsb_view1
WHERE 专业='计算机应用技术' AND 性别='女';
```

本例的运行结果如图 9.30 所示。

学号	姓名	专业	性别	出生日期	总分数
14001	马芳	计算机应用技术 ...	女	1989-8-16	89
14002	宋静	计算机应用技术 ...	女	1990-5-12	83

图 9.30　视图 xsb_view1 的查询结果

【例 9.12】　对视图 CJB_VIEW1 进行查询，查询课程号为“101”成绩在 87 分或 87 分以上的学生的学号和成绩（**实例位置：光盘\TM\sl\9\10**）。

```
SELECT 学号,成绩 FROM mr.cjb_view1
WHERE 成绩>=87;
```

本例运行结果如图 9.31 所示。

学号	成绩
14001	89
14003	92
14005	88
14006	92

图 9.31　成绩>=87 的查询结果

9.4 更新视图

更新视图是指通过视图来插入、修改和删除数据。由于视图是不实际存储数据的虚拟表，因此对视图的更新最终要转换成对数据表的更新。

为了防止用户通过视图对数据进行增加、修改和删除，对于不属于视图范围内的基本表数据进行操作，可在定义视图时加上 WITH CHECK OPTION 子句。这样在视图上进行增加、修改和删除操作时，系统就会检查视图定义中的条件，若不满足条件，则拒绝执行。

通过更新视图（包括插入、修改和删除）数据可以修改基本表数据。但并不是所有的视图都可以更新，只有满足可更新条件的视图，才能进行更新，因此必须保证要更新的视图是一个可更新的视图，同时，要注意基本表对数据的各种约束和规则要求。

9.4.1 使用界面方式操作视图数据

【例 9.13】 通过“控制中心”操作（包括插入、修改和删除记录）MRKJ 数据库中的 XSB_VIEW1 视图。

实现步骤如下：

（1）在“控制中心”对象视图中，选择 MRKJ 数据库下的“视图”文件夹，在内容窗格中选择“XSB_VIEW1”，然后单击鼠标右键，在弹出的快捷菜单中选择“打开”命令，如图 9.32 所示。

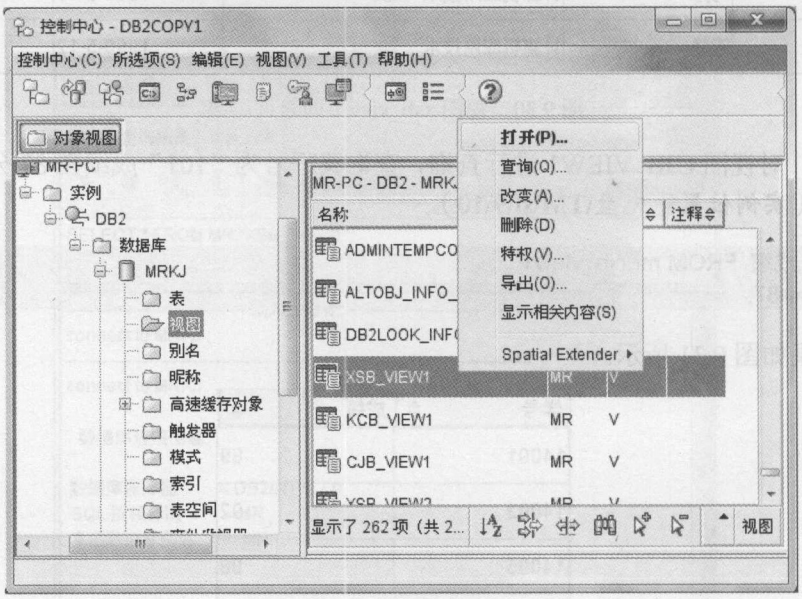


图 9.32 打开视图菜单

（2）在弹出的“打开视图”对话框中，视图中的记录按行显示，每个记录占一行。在此界面中，

可向视图中插入记录，也可以删除和修改记录，基本操作方式与对基本表的操作方式是一样的，如图 9.33 所示。

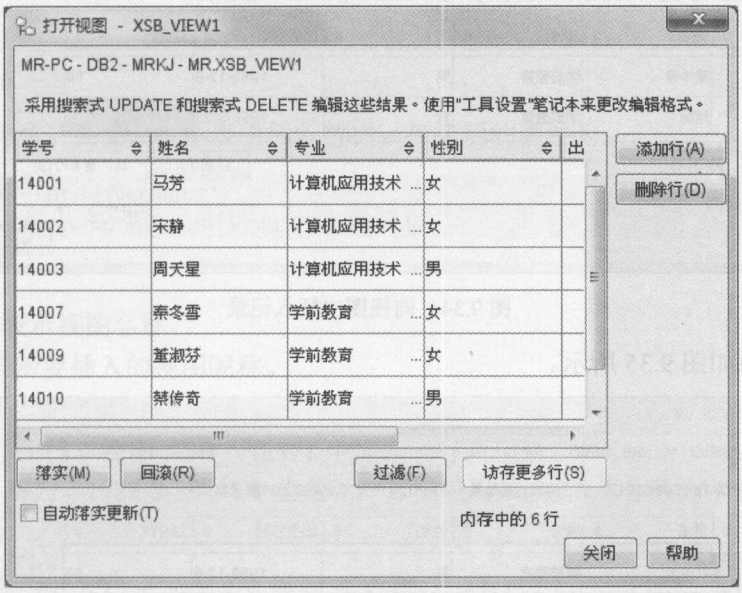


图 9.33　“打开视图”对话框

1. 插入记录

在视图中插入记录与对基本表插入记录基本一致，将新记录添加在视图尾，也可以向视图中插入多条记录。

操作方法是：将游标定位到当前视图尾的一行，单击“添加行”按钮，在新行中逐列输入列的值。每输入完一列的值，使用鼠标单击到其他列，便可编辑该列。若当前列是表的最后一列，再次单击“添加行”按钮，游标将自动跳到下一行的第一列，此时便可编辑新行。

输入完成后，此时输入信息并没有存到数据库，还必须单击“落实”按钮，输入的信息才会提交存盘。也可以在对话框左下方选中“自动落实更新”复选框，就不需要手工单击“落实”按钮来存盘了，系统会自动提交存盘。

【例 9.14】　向学生信息视图 XSB_VIEW1 中插入一条新的学生记录。学号为“14011”，姓名为“范婉”，性别为“女”，出生日期为“1990-3-23”，专业是“学前教育”，总分数为“63”。插入记录的页面如图 9.34 所示。

2. 删除记录

通过视图，用户也可以删除其中的记录。同样在“打开表”对话框中，选择需要删除的记录，单击“删除行”按钮，记录在界面中即消失。如果删错行或者不想删除了，单击“回滚”按钮，回滚刚才操作，消失的记录会返回界面。如果确认无误，继续删除，单击“落实”按钮来提交存盘。

【例 9.15】　删除学生信息视图 XSB_VIEW1 中学号为“14011”的学生信息。

打开视图 XSB_VIEW1，选中学号为“14011”的记录，然后单击“删除行”按钮，再单击“落实”按钮，最后单击“关闭”按钮完成删除操作。

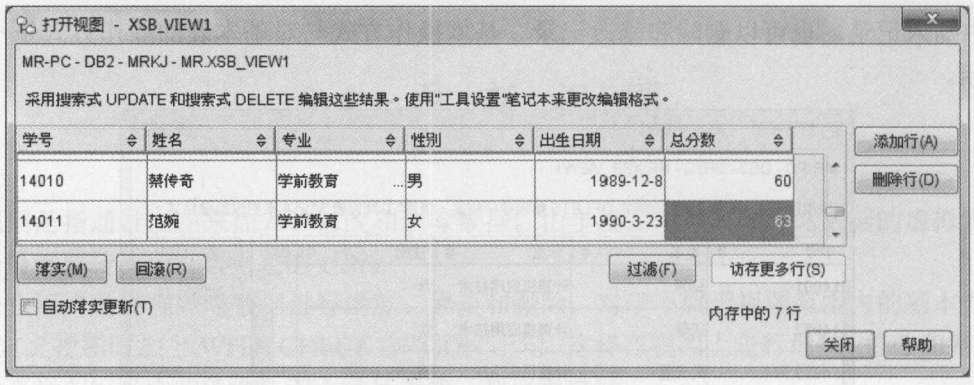


图 9.34　向视图中插入记录

删除记录的界面如图 9.35 所示。

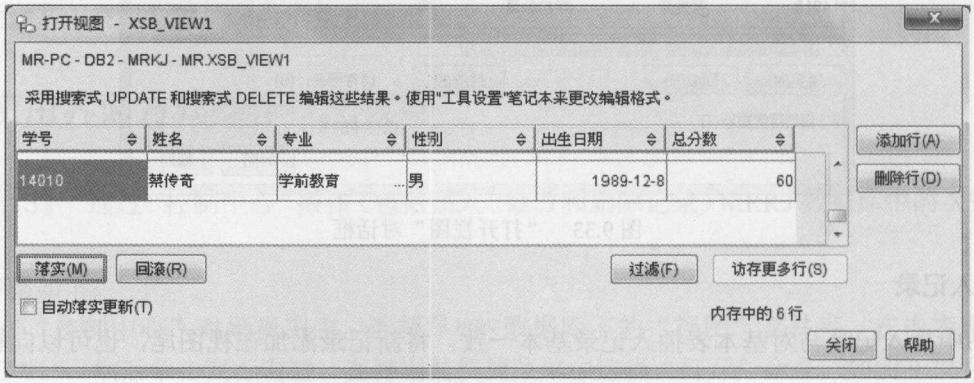


图 9.35　删除视图中的记录

3. 修改记录

修改记录数据的方法和修改基本表数据的方法是一样的，先定位要修改的记录字段，然后对该字段值进行修改，再单击“落实”按钮，最后单击“关闭”按钮完成删除操作。

【例 9.16】　将学生信息视图 XSB_VIEW1 中学号为“14003”的“周天星”的总分数更改成“67”分。修改记录的界面如图 9.36 所示。

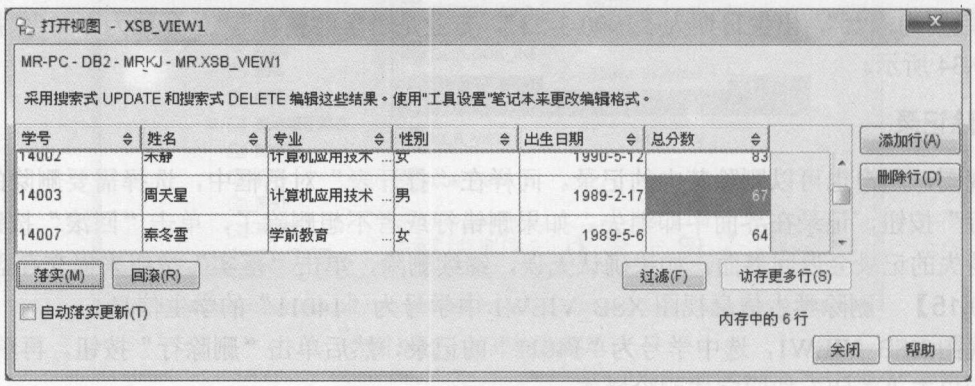


图 9.36　通过视图修改表的记录

9.4.2　使用命令方式操作视图数据

1. 插入记录

使用 INSERT 语句通过视图向基本表中插入数据。简化后的语法为：

```
INSERT INTO < view-name | nickname>
    <fullselect | [({column-name}[,…n])] VALUES values-list>
```

参数说明如下。

☑ view-name：表示视图名称。

☑ nickname：表示要插入的视图昵称。

☑ fullselect：是全查询语句。

【例 9.17】　向学生信息视图 XSB_VIEW1 中插入一行新记录。学号为“14969”，姓名为“赵亮”，性别为“男”，出生日期为“1989-8-30”，专业是“计算机应用技术”，总分数为“50”分（**实例位置：光盘\TM\sl\9\11**）。

```
INSERT INTO mr.xsb_view1 (学号,姓名,专业,性别,出生日期,总分数)
VALUES ('14969', '赵亮', '计算机应用技术', '男', '1989-8-30',50);
```

然后使用 SELECT 语句查询视图 XSB_VIEW1 依据的基本表“学生表”：

```
SELECT * FROM mr.学生表;
```

添加后的结果如图 9.37 所示。

学号	姓名	专业	性别	出生日期	总分数	照片	联系方式	备注
14005	董敏	计算机应用技术	女	1992-8-5	88		XML	
14006	万方	计算机应用技术	女	1993-8-12	92		XML	
14007	秦冬雪	学前教育	女	1989-5-6	64		XML	
14008	欧阳云	学前教育	男	1992-12-13	68		XML	
14009	董湘芬	学前教育	女	1989-8-16	62		XML	
14010	禁传奇	学前教育	男	1989-12-8	60		XML	
14969	赵亮	计算机应用技术	男	1989-8-30	50		XML	

图 9.37　向表中添加记录

2. 删除记录

使用 DELETE 语句可以通过视图删除基本表中的数据，简化后的语法为：

```
DELETE FROM view-name
WHERE search-condition
```

参数说明如下。

☑ view-name：视图的名称。

☑ search-condition：条件表达式。

注意

对于依赖多个基本表的视图不能使用 DELETE 语句。

【例 9.18】 从例 9.3 创建的“CJB_VIEW1”视图中删除学号为“14006”的学生记录（**实例位置：光盘\TM\sl\9\12**）。

```
DELETE FROM mr.cjb_view1
WHERE 学号='14006';
```

使用 SELECT 语句查询视图 CJB_VIEW1 依赖的基本表“成绩表”：

```
SELECT * FROM mr.成绩表
```

将会发现表中学号为“14006”的学生记录已经不存在了，如图 9.38 所示。

学号	课程号	成绩
14001	101	89
14002	101	83
14003	101	92
14004	101	84
14005	101	88
14007	102	64
14008	102	68
14009	102	62
14010	102	60

图 9.38 通过视图删除表中的记录

3. 修改记录

使用 UPDATE 语句可以通过视图修改基本表中的数据。

【例 9.19】 将例 9.3 创建的 CJB_VIEW1 视图中学号为“14005”的学生的成绩改成“97”分（**实例位置：光盘\TM\sl\9\13**）。

```
UPDATE mr.cjb_view1
SET 成绩=97
WHERE 学号='14005';
```

然后使用 SELECT 语句查询视图 CJB_VIEW1 依赖的基本表“成绩表”：

```
SELECT * FROM mr.成绩表 WHERE 学号='14005';
```

将会发现学号“14005”的学生成绩已经修改成“97”，如图 9.39 所示。

学号	课程号	成绩
14005	101	97

图 9.39　通过视图修改表记录

9.5　修改视图的注释

视图定义后只能修改注释，除注释以外的其他属性都不能修改。如果要修改视图的注释以外的属性，只能删除视图并重新创建视图。

有两种方式可以添加注释：一种是在界面中为视图添加注释，另一种是使用 SQL 命令为视图添加注释。

9.5.1　使用界面方式为视图 CJB_VIEW1 添加注释

在“控制中心”的对象视图中选择 MRKJ 数据库下的“视图”文件夹，在内容窗格中找到“CJB_VIEW1”，单击鼠标右键，在弹出的快捷菜单中选择“改变”命令，如图 9.40 所示，在弹出的“改变视图”对话框中的“注释”文本框中添加注释“计算机应用技术学生成绩视图”，如图 9.41 所示。

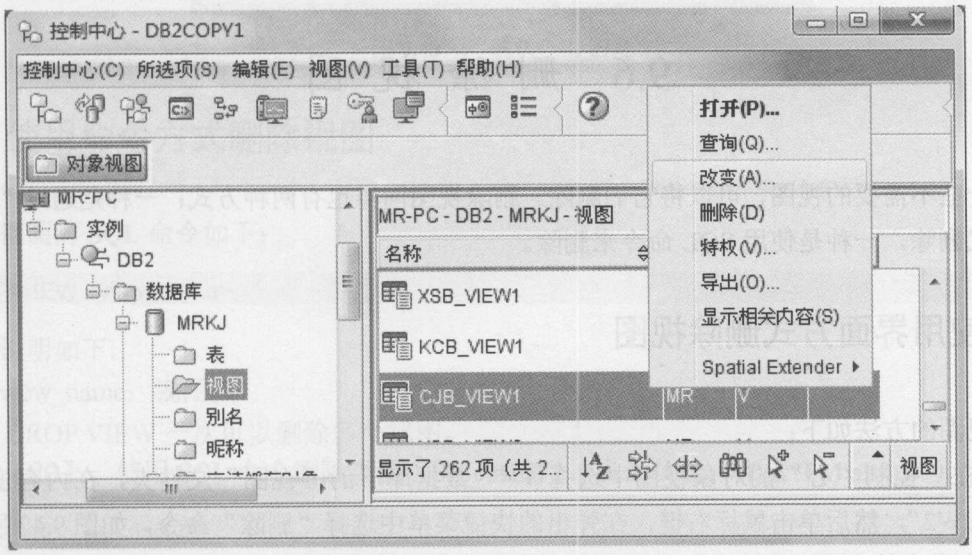


图 9.40　修改视图注释

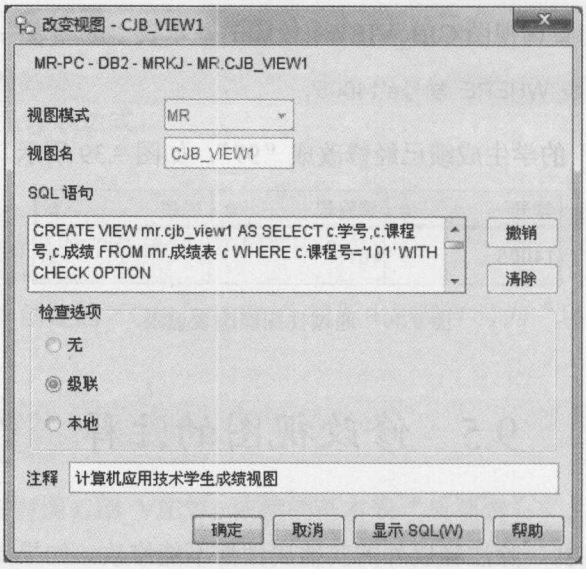


图 9.41　修改视图的注释

9.5.2　使用命令方式为视图添加注释

使用命令方式为视图添加注释的语法如下：

```
COMMENT ONTABLE mr.cjb_view IS '计算机应用技术学生成绩视图';
```

想要删除视图的注释，可用下面的 SQL 语句：

```
COMMENT ON TABLE mr.cjb_view1 IS;
```

9.6　删 除 视 图

对于一些不需要的视图，可以将它们删除。删除视图同样也有两种方式：一种是通过“控制中心”的界面方式删除，一种是使用 SQL 命令来删除。

9.6.1　使用界面方式删除视图

删除视图的方法如下：

（1）在“控制中心”的对象视图中选择 MRKJ 数据库下的“视图”文件夹，在内容窗格中找到“XSB_VIEW2”，然后单击鼠标右键，在弹出的快捷菜单中选择“删除”命令，如图 9.42 所示。

（2）在弹出的“确认”对话框中，确认无误后单击“确定”按钮，视图即被删除，如图 9.43 所示。

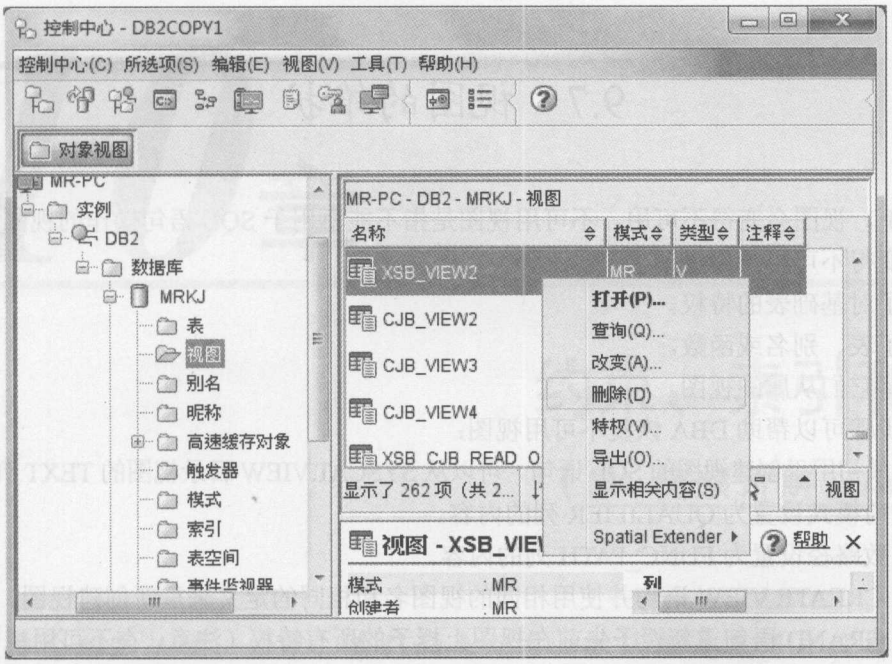


图 9.42　删除视图菜单

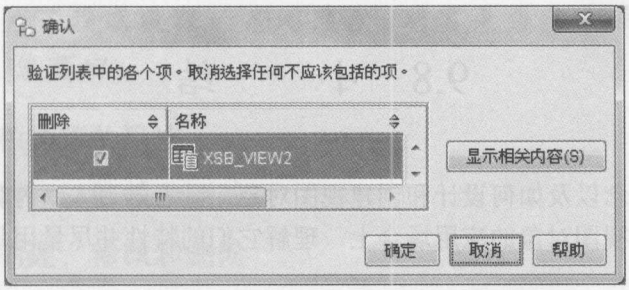


图 9.43　“确认”对话框

9.6.2　使用命令方式删除视图

删除视图的 SQL 命令如下：

```
DROP VIEW <view_name>
```

参数说明如下。

☑　view_name：视图名。

使用 DROP VIEW 一次可以删除多个视图。

【例 9.20】　使用 SQL 命令删除视图 XSB_VIEW2（**实例位置：光盘\TM\sl\9\14**）。

```
DROP VIEW xsb_view2;
```

9.7 视图的维护

有些情况下，视图会变得不可用，不可用视图是指不能再用于 SQL 语句操作的视图。在下列情况下，视图可能变得不可用：

☑ 撤销了对基础表的特权。

☑ 删除了表、别名或函数。

☑ 删除了它们从属的视图。

通过以下步骤可以帮助 DBA 恢复不可用视图：

（1）确定最初用于创建视图的 SQL 语句。可以从 SYSCAT.VIEW 目录视图的 TEXT 列获取此信息。

（2）将当前模式设置为 QUALIFIER 列的内容。

（3）将函数路径设置为 FUNC_PATH 列的内容。

（4）使用 CREATE VIEW 语句并使用相同的视图名和相同的定义来重新创建视图。

（5）使用 GRAND 语句重新授予先前在视图上授予的所有特权（注意，在不可用视图上授予的所有特权都会被撤销）。

9.8 小　　结

本章介绍了视图的概念以及如何设计和创建视图对象，读者学习本章内容时，关注点不应该放到创建命令上，而应该放到视图对象的应用场合上，理解它们的特性并尽量用最合理的数据库技术来实现业务需求。

9.9 实践与练习

（1）视图与表是什么关系？视图是实际表还是虚拟表？（**答案位置：光盘\TM\sl\9\15**）

（2）哪些视图不允许更新？（**答案位置：光盘\TM\sl\9\16**）

（3）可以修改视图的定义吗？（**答案位置：光盘\TM\sl\9\17**）

第10章

约束、索引和别名

（视频讲解：46分钟）

本章主要介绍默认值、约束、索引和表别名。默认值是在用户没有明确指定某列值的情况下，系统自动填充预先定义或指定的值；约束是为了使信息符合特定的限制或规则；索引是为了提高数据的查阅速度；别名是为了方便对表或视图的操作。

通过阅读本章，您可以：

- 掌握默认值和约束的用法
- 掌握索引的创建、修改和删除
- 掌握别名的创建、修改和删除

10.1 默认值和约束

数据通常必须符合特定限制或规则，这些限制可能适用于单条信息（例如，格式和序号），也可能适用于若干条信息，下面是一些常用的限制规则。

1. 列数据值的可空性

空值表示未知状态。默认情况下，所有内置数据类型都支持空值的存在。但是，一些业务规定可能要求必须始终为某些列提供值，例如，员工编号。对于这种情况，可以使用 NOT NULL 约束来确保始终不会为给定表列指定空值。为特定列定义 NOT NULL 约束后，尝试在该列中放入空值的任何插入或更新操作都将失败。

2. 默认列数据值

正如一些业务规则要求必须始终提供值一样，其他业务规则可能要求该值应该是什么，例如，“学生表”中的“性别”必须是“男”或“女”。列默认值约束用于确保在表中添加给定表列没有特定值的行时，始终为该列指定预定义的值。为列提供的默认值可以是空值、与该列的数据类型兼容的约束值或数据库管理器提供的值。

3. 键

键是表或索引中可用来标识或访问特定数据行的单列或一组列。任何列都可以是键的一部分，并且同一列可以是多个键的一部分。

☑ 由单列组成的键称为“原子键”；由多列组成的键称为“组合键”。

☑ 除了具有原子或组合属性外，还根据使用键实施约束的方式对键进行了分类：

（1）“唯一键”用来实施唯一约束。

（2）“主键”用来实施实体完整性约束（主键是一种特殊的唯一键，它不支持空值）。

（3）“外键”用来实施引用完整性约束（外键必须引用主键或唯一键，外键没有相应的索引）。

通常在声明表、索引或引用约束定义期间指定键。

4. 完整性约束

约束是对可在表中插入、删除或更新的值进行限制的规则。

约束包括非空约束、检查约束、唯一约束（也称为唯一键约束）、主键约束、外键约束（也称为引用约束、引用完整性约束或参照约束）。

10.1.1 默认值

在表中新增记录时，如果用户没有明确指定某些列的值，在这种情况下，由系统自动填充的为某些列和数据类型预先定义或指定的值，称为“默认值”。

各种数据类型的默认值如下。

（1）NULL：Char、VarChar 等字符串类型。

（2）0：用于小整数、整数、十进制、单精度浮点数和双精度浮点数。

（3）空白：用于固定长度字符串和固定长度双字节字符串。

（4）零长度字符串：用于变长字符串、二进制大对象、字符大对象和双字节字符大对象。

（5）日期：这是插入行时的系统日期（从 CURRENT_DATE 专用寄存器获取）。将日期列添加至现有表时，将为现有行指定日期 2001 年 01 月 01 日。

（6）时间或时间戳记：这是插入语句时的系统时间或系统日期/时间（从 CURRENT_TIME 专用寄存器获取）。将时间列添加至现有表时，将为现有行指定时间 00:00:00 或包含日期 0001 年 01 月 01 日和时间 00:00:00 的时间戳记。

注意

所有行将获得给定语句的相同默认时间/时间戳记值。

（7）用户定义的单值数据类型：这是系统为用户定义的单值数据类型的基本数据类型定义的默认值（强制类型转换为用户定义的单值数据类型）。

10.1.2　约束

约束只与表关联，它们是在创建表的过程中定义的（使用 CREATE TABLE 语句），或者是在创建表后添加至表定义的（使用 ALTER TABLE 语句）。可以使用 ALTER TABLE 语句来修改约束。在大多数情况下，随时都可以删除现有约束；此操作不会影响表结构和存储在表中的数据。

1. 创建和修改约束

可以使用 ALTER TABLE 语句将约束添加至现有表。

约束名不能与在 ALTER TABLE 语句内指定的任何其他的约束相同，且必须在该表内是唯一的（这包括定义的任何引用完整性约束的名称）。在成功执行该语句之前，会对照新条件检查现有数据。

要修改此约束，必须先删除此约束，然后重新创建。

在某个列上定义约束可以有 4 种方式：

☑　界面上创建表时添加约束。

☑　SQL 命令创建表时添加约束。

☑　创建表后在界面中添加约束。

☑　创建后用 SQL 语句添加约束。

创建和修改唯一约束的方法和步骤如下：

可以将唯一约束添加至现有表。约束名不能与在 ALTER TABLE 语句内指定的任何其他约束相同，且必须在该表内是唯一的（这包括定义的任何引用完整性约束的名称）。在成功执行该语句之前，会对照新条件检查现有数据。

使用 ALTER TABLE 语句的 ADD CONSTRAINT 选项来定义唯一约束。

【例 10.1】 在“学号”上创建一个唯一约束（**实例位置：光盘\TM\sl\10\1**）。

```
ALTER TABLE mr.学生表
    DROP PRIMARY KEY
            --首先要删除“学号”上已有的主键，才能重新创建“学号”上的唯一约束
    ADD CONSTRAINT xsb_unique_key_name UNIQUE(学号)
            --在“学号”上添加唯一约束，约束名为 xsb_unique_key_name
```

本例运行结果如图 10.1 所示。

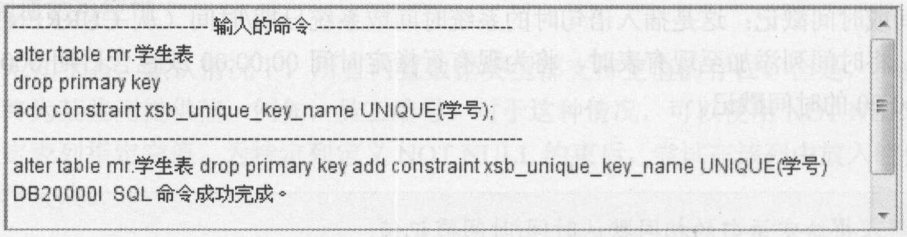


图 10.1 添加“唯一约束”

说明

“学生表”中“学号”列已经有主键约束了，不能再创建唯一约束，否则系统会提示错误。但可以先将“学生表”中定义在“学号”上的主键删除，并在“学号”上重新创建一个唯一约束，它表示唯一标识该表中的学号的一个新方法。

创建和修改主键约束的方法和步骤如下：

可以将主键约束添加至当前表中。约束名必须在表内是唯一的（这包括定义的任何引用完整性约束的名称）。在成功执行该语句之前，会对照新条件检查现有数据。

创建主键约束的语句语法为：

```
ALTER TABLE table_name
    ADD CONSTRAINT primary_key_name
    PRIMSRY KEY { column_name }[,...n];
```

不能修改现有主键约束。要将另一列或另一组列定义为主键，必须先删除现有主键定义，然后再重新创建。

创建和修改检查约束的方法和步骤如下：

当添加表检查约束时，插入或更新该表的程序包和高速缓存的动态 SQL 可能被标记为无效。

创建检查约束的语句语法为：

```
ALTER TABLE table_name
    ADD CONSTRAINT check_name CHECK (logical_expression)
    [NO ENFORCED | ENFORCED]
    [ENABLE QUERY OPRIMIZATION];
```

【例 10.2】 假如“学生表”的“性别”列上还没有约束，在“学生表”的“性别”列上添加约束，“性别”只能为“男”或“女”（本实例的约束已经添加，不能再添加相同约束，读者可以将约束

删除，然后重新用 SQL 语句添加）（**实例位置：光盘\TM\sl\10\2**）。

```
ALTER TABLE mr.学生表
    ADD CONSTRAINT xsb_check CHECK (性别 = '男' OR 性别 ='女')
            --性别只能是“男”或“女”
    ENFORCED   --强制此约束
    ENABLE QUERY OPTIMIZATION;   --在查询优化期间考虑此约束
```

本例运行结果如图 10.2 所示。

```
------------------------------ 输入的命令 ------------------------------
alter table mr.学生表
add constraint xsb_check check(性别='男' OR 性别='女')
enforced
enable query optimization;
------------------------------------------------------------------------
alter table mr.学生表 add constraint xsb_check check(性别='男' OR 性别='女') enforced enable query optimization
DB20000I  SQL 命令成功完成。
```

图 10.2　检查约束

【例 10.3】　在课程表上的“开课学期”上添加约束——“开课学期”必须是 1~8（这个约束实际上已经添加过了，不能再次添加相同的约束，想要添加一定要删除以前添加过的约束）（**实例位置：光盘\TM\sl\10\3**）。

```
ALTER TABLE mr.课程表
    ADD CONSTRAINT kcb_check CHECK (开课学期>0 AND 开课学期<9)
        --开课学期只能为 1~8
    ENFORCED    --强制此约束
    ENABLE QUERY OPTIMIZATION;
        --在查询优化期间考虑此约束
```

本例运行结果如图 10.3 所示。

```
------------------------------ 输入的命令 ------------------------------
alter table mr.课程表
add constraint kcb_check check (开课学期>0 AND 开课学期<9)
ENFORCED
ENABLE QUERY OPTIMIZATION;
------------------------------------------------------------------------
alter table mr.课程表 add constraint kcb_check check (开课学期>0 AND 开课学期<9) ENFORCED ENABLE QUERY OPTIMIZATION
DB20000I  SQL 命令成功完成。
```

图 10.3　检查约束

创建和修改外键约束的方法和步骤如下：

外键约束（也称为引用约束或引用完整性约束）能够定义表间以及表内必须的关系。外键是表中的一列或一组列，要求其值与其父表或昵称中某行的至少一个主键值相匹配。

创建外键（引用）约束的语句语法为：

```
ALTER TABLE table_name
    ADD CONSTRAINT foreign_key_constraint_name
    FOREIGN KEY ({column_name} [,...n])
```

```
    REFERENCES
    parent_table_name ({column_name} [,...n])
    ON DELETE delete_action_type
    ON UPDATE update_action_type
    [ENFORCED | NOT ENFORCED]
    [ENABLE QUERY OPTIMIZATION | DISABLE QUERY OPTIMIZATION];
```

其中：

```
delete_action_type ::=
NO ACTION | RESTRICT | CASCADE | SET NULL
update_action_type ::=
NO ACTION | RESTRICT
```

一个表可以同时创建多个外键约束。要修改此约束，必须先删除此约束，然后再重新创建。

数据库管理器通过强制执行与每个引用约束关联的一组规则来处理这类情况。这组规则包括：插入规则、更新规则、删除规则，下面分别进行介绍。

（1）插入规则。引用约束的插入规则为：外键的非空插入值必须与父表的父键的某些值相匹配。如果组合外键的值的任何组成部分为空，那么该值为空。指定外键时，此规则是隐式的。

（2）更新规则。引用约束的更新规则是在定义引用约束时指定的。选项有 NO ACTION 和 RESTRICT。在更新父表的某行或从属表的某行时应用更新规则。

如果是父行，更新父键的某列中的值时，下列规则适用：

☑ 如果从属表中的任何行与该键的原始值相匹配，在更新规则为 RESTRICT 的情况下，会拒绝更新。

☑ 如果在更新语句完成时，从属表中的任意行没有相应的父键（排除 AFTER 触发器），当更新规则为 NO ACTION 时，会拒绝更新。

如果是从属行，当指定外键时，NO ACTION 更新规则是隐式的。NO ACTION 意味着更新语句完成时，外键的非空更新值必须与父表的父键的某些值相匹配。

如果组合外键的值的任何组成部分为空，那么该值为空。

（3）删除规则。引用约束的删除规则是在定义引用约束时指定的。选项有 NO ACTION、RESTRICT、CASCADE 或 SET NULL。只有当外键的某些列允许空值时，才能指定 SET NULL。

在删除父表的某行时应用引用约束的删除规则。更确切地说，当父表的某行是删除或传播删除操作的对象，且该行在引用约束的从属表中具有从属项时，该规则适用。考虑这样一个例子，其中 P 是父表，D 是从属表，而 P 是充当删除或传播删除操作的对象的父行。删除规则的工作方式如下：

☑ 对于 RESTRICT 或 NO ACTION，发生错误时不会删除任何行。

☑ 对于 CASCADE，删除操作会传播至表 D 中的 P 的从属项。

☑ 对于 SET NULL，表 D 中的 P 的每个从属项的外键的每个可空列被设置为空。

【例 10.4】 在“成绩表”上添加两个关键字 cjb_foreign_key1 和 cjb_foreign_key2（**实例位置：光盘\TM\sl\10\4**）。

```
ALTER TABLE mr.成绩表
ADD CONSTRAINT cjb_foreign_key1
    FOREIGN KEY (学号) REFERENCES mr.学生表(学号)
        ON DELETE CASCADE
```

```
        ON UPDATE RESTRICT
        ENFORCED
        ENABLE QUERY OPTIMIZATION
        ADD CONSTRAINT cjb_foreign_key2
FOREIGN KEY(课程号) REFERENCES mr.课程表(课程号)
        ON DELETE RESTRICT
        ON UPDATE NO ACTION
        ENFORCED
        ENABLE QUERY OPTIMIZATION;
```

本例运行结果如图 10.4 所示。

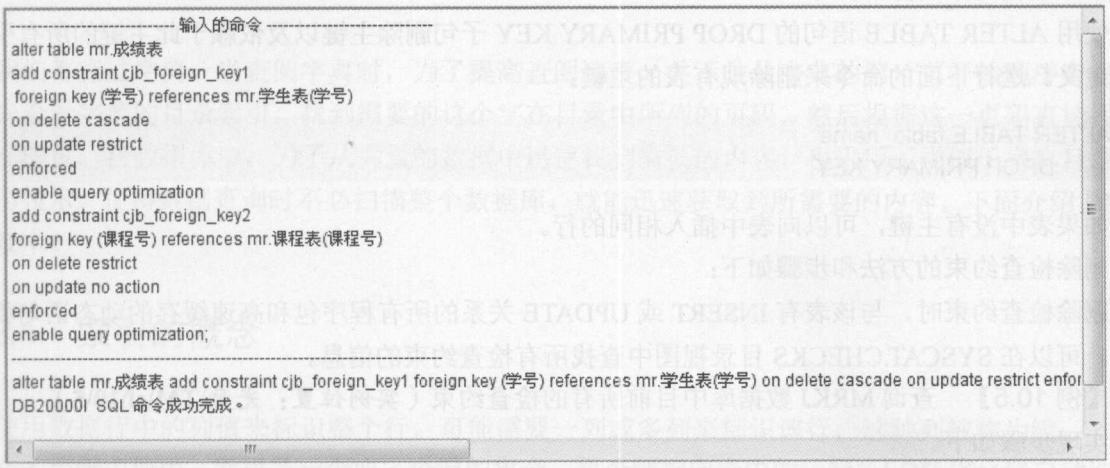


图 10.4　添加外键

说明

（1）外键关键字 cjb_foreign_key1 将学生表看成“父表”，将成绩表看成“从属表”；插入规则是隐式的，如果要向“成绩表”中插入一条记录，那么这条成绩记录中的“学号”的值必须是“学生表”中已经存在的某个“学号”；删除规则为 CASCADE，当删除父表“学生表”中的一条学生记录时，系统自动删除从属表“成绩表”中相同学号的学生成绩记录；更新规则为 RESTRICT，如果在尝试更新“学生表”中的父键“学号”时，父行（即“学生表”中准备更新的行）有任何从属行（即“成绩表”中依赖于“学生表”中即将更新的那些“学号”的行），那么不能更新父行中父键的值。

（2）外键关键字 cjb_foreign_key2 将课程表看成“父表”，将成绩表看成“从属表”；插入规则是隐式的，如果要向“成绩表”中插入一条记录，那么这条成绩记录中的“课程号”的值必须是“课程表”中已经存在的某个“课程号”；删除规则为 RESTRICT，如果父行在删除时有任何从属行，那么不能删除父表的行。更新规则为 NO ACTION，如果父行在语句完成时有任何从属行，那么不能更新父行中父键的值。

3．删除约束

可以使用含有 DROP 或 DROP CONSTRAINT 子句的 ALTER TABLE 语句显式删除表约束，或作

为 DROP TABLE 语句的结果将其隐式删除。

删除唯一约束的方法如下。

使用 ALTER TABLE 语句的 DROP UNIQUE 子句，删除现有唯一约束以及依赖于此唯一约束的所有引用约束的定义。删除此唯一约束使引用该约束的任何程序包或高速缓存的动态语句无效。运行下面命令来删除表的唯一约束 unique_constraint_name：

```
ALTER TABLE table_name
    DROP UNIQUE unique_constraint_name;
```

删除主键约束的方法如下。

使用 ALTER TABLE 语句的 DROP PRIMARY KEY 子句删除主键以及依赖于此主键的所有引用约束的定义。运行下面的命令来删除现有表的主键：

```
ALTER TABLE table_name
    DROP PRIMARY KEY;
```

如果表中没有主键，可以向表中插入相同的行。

删除检查约束的方法和步骤如下：

删除检查约束时，与该表有 INSERT 或 UPDATE 关系的所有程序包和高速缓存的动态语句将变得无效。可以在 SYSCAT.CHECKS 目录视图中查找所有检查约束的信息。

【例 10.5】 查询 MRKJ 数据库中目前所有的检查约束（**实例位置：光盘\TM\sl\10\5**）。

实现步骤如下。

（1）连接到 MRKJ 数据库：

```
CONNECT TO mrkj USER mr USING 1;
```

（2）查询 MRKJ 数据库中目前所有的检查约束：

```
SELECT * FROM SYSCAT.CHECKS;
```

运行结果如图 10.5 所示。

CONSTNAME	OWNER	OWNERTYPE	TABSCHEMA	TABNAME	CREATE_TIME	QUALIFIER	TYPE
XSB_CHECK	MR	U	MR	学生表	2014-8-15 15:23...	MR	C
KCB_CHECK	MR	U	MR	课程表	2014-8-15 15:59...	MR	C

图 10.5 查看所有“检查约束”

删除检查约束的语句的语法为：

```
ALTER TABLE table_name
    DROP CONSTRAINT check_constraint_name;
```

删除外键（引用）约束的方法如下。

使用 ALTER TABLE 语句的 DROP FOREIGN KEY 子句删除现有外键约束。运行以下命令来删除现有外键约束 foreign_key_name：

```
ALTER TABLE table_name
DROP   FOREIGN KEY foreign_key_name;
```

当删除外键约束时，包含下列语句的程序包或高速缓存的动态语句可能被标记为无效：

☑ 插入或更新包含外键表的语句。

☑ 更新或删除父表的语句。

10.2 索　　引

我们都查过字典，当查阅字典时，为了提高查阅速度，并不是从字典的第一页开始顺序查找，而是首先查看字典的目录索引，找到需要的这个字在目录中所列的页码，然后根据这一页码直接找到该字所在的页。在数据库中，为了从大量的数据中迅速找到需要的内容，也采用了类似于字典目录这样的索引技术，使得数据查询时不必扫描整个数据库，就能迅速获取到所需要的内容。下面介绍 DB2 的索引技术。

10.2.1 索引的概念

使用数据行中的列值来标识整个行，可能需要一列或多列来标识该行，这种列被称为键。一个列可以在多个键中使用。索引是一个或多个键的集合，每个键指向表中的一行。SQL 优化器自动选择最有效率的访问表中数据的方法。当确定最快速的数据访问路径时，优化器会将索引考虑在内。

在数据库系统中建立索引主要有以下作用：

（1）快速存取数据。

（2）保证数据记录的唯一性。

（3）实现表与表之间的参照完整性。

（4）在使用 ORDER BY、GROUP BY 子句进行数据检索时，利用索引可以减少排序和分组的时间。

将数据添加到表时，除非已对该表和正在添加的数据执行了其他操作，否则简单地将数据追加至该表的底部。数据是无序的，搜索特定数据行时，必须检查从第一行到最后一行的每个表行。将索引用作按顺序访问表中数据的一种方法，该顺序在其他情况下可能不可用。

索引按键中的值进行排序。键可以是唯一的，也可以是非唯一的。每个表应该至少有一个唯一键；但还可以有其他唯一键。每个索引正好有一个键，例如，可以使用员工编号（唯一）作为一个索引键，并使用部门号（非唯一）作为另一个索引键。

索引是根据表中一列或若干列按照一定顺序建立的列值与记录行之间的对应关系表。

设计索引的原则如下：

（1）在创建索引时要记住，虽然它们可以提高读取性能，但会对写性能产生负面影响。这是因为对于数据库管理器写入表中的每行，它还必须更新任何受影响的索引。因此，只有在会明显提高整体性能时，才应创建索引。

（2）在创建索引时，还必须考虑表结构和经常对这些表执行的查询的类型。例如，经常发出查询

的 WHERE 子句中出现的列是索引的优秀候选者，但是，在较少运行的查询中，索引对 INSERT 和 UPDATE 语句的性能产生的负面影响可能超过其所带来的好处。

（3）同样，在经常运行的查询的 GROUP BY 子句中出现的列可能会从创建索引中获益，尤其在用于分组行的值的数目小于要分组的行数时。

10.2.2 索引的分类

可以按照两种分类角度给索引分类：按照唯一性分类，可分为唯一索引、非唯一索引；按照索引的键顺序与数据的物理存储顺序是否一致，可分为集群索引和非集群索引。

1. 唯一索引和非唯一索引

唯一索引是这样的一种索引，它通过确保表中没有两个数据行具有完全相同的键值来帮助维护数据完整性。

尝试为已经包含数据的表创建唯一索引时，将检查组成该索引的列中的值是否唯一；如果该表包含具有重复键值的行，那么索引创建过程将失败。为表定义唯一索引后，将在插入和更新操作期间强制唯一性。除了强制数据值的唯一性外，唯一索引还可以用来提高查询处理期间检索数据的性能。

另外，非唯一索引不用于对与它们关联的表强制执行约束。相反，非唯一索引通过维护频繁使用的数据值的排序顺序，仅仅用于提高查询性能。

2. 集群索引和非集群索引

索引体系结构分为集群或非集群。集群索引是根据数据在磁盘上排序所依据的那个键构建的索引。在某些关系数据库管理系统中，集群索引的叶子节点对应于实际数据，而不是对应于指向位于其他地方的数据的指针，非集群索引的叶子节点就对应于这种指针。

由于集群索引的叶子级别对应于实际数据这个事实，表中的数据将根据索引进行排序，因此，给定表中只能存在一个集群索引，但可以存在许多非集群索引。

非集群索引是在任何键的基础上构建的索引。每种关系都可以有一个集群索引和许多个非集群索引。集群索引通常将实际记录存储在数据结构中，因此其速度比非集群索引要快很多。非集群索引被强制仅将记录标识存储在数据结构中，并且需要至少一个其他 I/O 操作才能检索到实际记录。

重组与所选索引相关的表将对数据重新建立集群。建立了集群的索引对于带有范围谓词的列非常有用，因为它允许对表中的数据作更有效的顺序访问。由于相似的值在同一数据页上，从而减少了页访问次数。

通常，表中只有一个索引可以具有较高的集群度。

10.2.3 创建索引

相同列上相同排序方式的索引被认为是匹配的。如果要在主键所在的列上创建索引，必须先删除主键约束，才能重新在主键上创建索引。

如果要在“课程表”上按“课程名”建立唯一索引，索引组织方式为非集群索引，那么在 DB2 中，

可利用“控制中心”创建上述索引，也可以利用 SQL 命令通过“命令编辑器”建立索引。

注意

在创建“学生表”“课程表”“成绩表”时已经创建了主键约束，DB2 系统自动创建了分别与这 3 个表的主键约束名称一致的唯一索引“XSB_PRIMARY_KEY”“KCB_PRIMARY_KEY”“CJB_PRIMARY_KEY”。此时不能再创建这些主键上的主键索引或唯一索引，即使索引名与系统已自动创建的唯一索引名不同也不行。

用命令创建表的索引的语法格式如下：

```
CREATE [UNIQUE] INDEX index-name
    ON <table-name | nickname>(column-name [ASC | DESC] )
    [NOT PARTITIONED]
    [IN tablespace-name]
    [SPECIFICATION ONLY]
    [INCLUDE (not-enforced-unique-column-name ) ]
    [xml-index-specification | CLUSTER | EXTEND USING index-extension-name
[(constant-expression)]]
[PECFREE 10 | PCTFREE<INTEGER1>]
[LEVEL2 PCTFREE<INTEGER2>]
[MINPCTUSED<INTEGER3>]
[ALLOW REVERSE SCANS | DISALLOW REVERSE SCANS]
[PAGE SPLIT<SYMMETRIC | HIGH | LOW>]
[COLLECT [[SAMPLED] STATISTICS]   [DETAILED]
```

参数说明如表 10.1 所示。

表 10.1　创建索引语法的参数说明

参　数	描　述
UNIQUE	为表或视图创建唯一索引（即不允许存在索引值相同的两行），如果创建索引时表已经包含了重复的关键字，则不能创建唯一索引
index-name	索引名，包含显式或隐式的模式名，模式名不能是 SYSIBM、SYSCAT、SYSFUN 或 SYSSTAT
table-name	必须是一个基本表、编目中描述的物理表或已声明的临时表。已声明的临时表的模式名必须是 SESSION
nickname	表的昵称
column-name	列名，该列将成为索引关键字的成员列。可以为索引指定多个列
ASC	索引文件按升序建立，默认排序方式
DESC	索引文件按降序建立
NOT PARTITIONED	指明索引必须搜索表的所有数据分区
table-name	数据分区的表名
tablespace-name	指定将创建的索引所处的表空间，这个子句只有在分区表上才被支持
SPECIFICATION ONLY	指明创建的索引将应用到昵称所指定的数据源表，只有指定了昵称时才指定 SPECIFICATION ONLY

续表

参　数	描　述
INCLUDE not-enforced-unique-column-name	指定添加到索引列（column-name 所指的那些列）集合的其他列（not-enforced-unique-columnd-name 所指的那些列）。这个子句中的任何列都不强制唯一。这些添加的列必须强制唯一的列区分开，不能在声明的临时表上使用这个子句。如果指定昵称时使用了 EXTEND USING if nickname is specified 或者是 XML 列上的索引都不能使用这个子句
CLUSTER	指明是在表上创建集群索引。一个表上只存在一个集群索引，因此如果 CLUSTER 用于定义表上的非集群索引时，CLUSTER 可以省略
EXTEND USING index-extension-name	给管理索引的索引扩展的命名。如果指定了这个子句则只能指定一个列名，这个列必须是结构数据类型或独立数据类型。index-extension- name 必须是编目中描述的索引扩展的名称。对于独立数据类型，这个列必须完全匹配索引扩展中的相应的关键字；对于结构数据类型，索引扩展中的相应关键字类型必须与列类型相同或者是列类型的超类。临时表不能用这个子句
constant-expression	指定索引扩展需要的所有参数
PCTFREE <integer1>	指定建立索引时每个索引页保留百分之多少作为空余空间<integer1>的值，可以是 0~99，第一个值添加到索引页时没有任何限制，其他值添加到索引页时，每个非叶子索引页要保留百分之 min（10，<integer1>）的空余空间。<integer1>默认值是 10
LEVEL2 PCTFREE <integer2>	指定建立索引时每个二级索引页保留百分之多少作为空余空间，这个值可以是 0~99。如果没有指定 LEVEL2 PCTFREE，则每个非叶子索引页中保留百分之 min（10，<integer1>）的空余空间。如果指定了 LEVEL2 PCTFREE，则二级中间索引页保留百分之<integer2>的空余空间，3 级或者更高级别的中间索引页保留百分之 min（10，<integer2>）的空余空间
MINPCTUSED <integer3>	指定叶子索引页是否在线合并，并指定叶子索引页中已用空间的最小的百分数的门槛值。如果从叶子索引页移除一个键后，叶子索引页中已用空间的百分数小于等于百分之<integer3>，系统就试图将该叶子索引页上剩余的键与相邻索引页上的那些键合并。如果这些索引页上有足够的空间，合并就被进行，且删除其中一个索引页。<integer3>的值可以是 0~99。为了具有较好的性能，推荐<integer3>采用小于等于 50 的值。指定这个选项会对更新和删除性能产生影响。对于 2 型索引，只有包含独占锁的表执行删除或更新操作时才进行合并，如果表上的独占锁不存在，在更新或删除操作期间，键就被标记为假删除，且不进行合并。最好使用 REORG INDEXES 的 CLEANUP ONLY ALL 选项来合并叶子索引页，而不是使用 CREATE INDEX 的 MINPCTUSED 选项。临时表不能使用 MINPCTUSED 子句
DISALLOW REVERSE SCANS	指明索引仅支持前向索引
ALLOW REVERSE SCANS	指明索引支持前向和反向搜索
PAGE SPLIT	指定索引分页动作，默认值是 SYMMETRIC
SYMMETRIC	指定按照索引的关键字将索引页对称地分开
HIGH/LOW	当向索引关键字对应的列插入值时遵循特定的模式，按照高效利用索引页空间的方式分割索引页。对索引关键字的子集，索引中最左的一列或多列必须包含相同的值，索引中最右一列或多列必须包含递增/递减的值
LOW	指定按照高效利用索引页空间的方式分割索引页
COLLECT STATISTICS	指明索引创建期间收集基本的索引统计信息
DETAILED	指明索引创建期间收集延伸的统计信息（CLUSTER FACTOR 和 PAGE_FETCH_PAIRS）
SAMPLED	指明编译延伸的索引统计信息时可以采样

注意

（1）相同列上相同排序方式的索引被认为是匹配的，不能创建相同列上相同排序方式的索引。

（2）如果指定了昵称（nickname），则不能指定 INCLUDE、xml-index-specification（XML 列上的索引）、CLUSTER、EXTEND USING、PCTFREE、MINPCTUSED、DISALLOW REVERSE SCANS、ALLOW REVERSE SCANS、PAGE SPLIT 或 COLLECT STATISTICS。

（3）只有指定了 UNIQUE 时才能用 INCLUDE 子句。

（4）如果指定了 xml-index-specification，则不能指定 column-name DESC、INCLUDE 或 CLUSTER。

【例 10.6】　在“课程表”上按“课程名”建立唯一索引，索引组织方式为非集群索引。

1. 通过控制中心建立表的索引

建立索引的方法如下：

（1）在“控制中心”对象视图中选择 MRKJ 数据库下的“表”文件夹，然后在内容窗格中找到 课程表，然后单击鼠标右键，在弹出的快捷菜单中选择“创建”→“索引”命令，如图 10.6 所示。

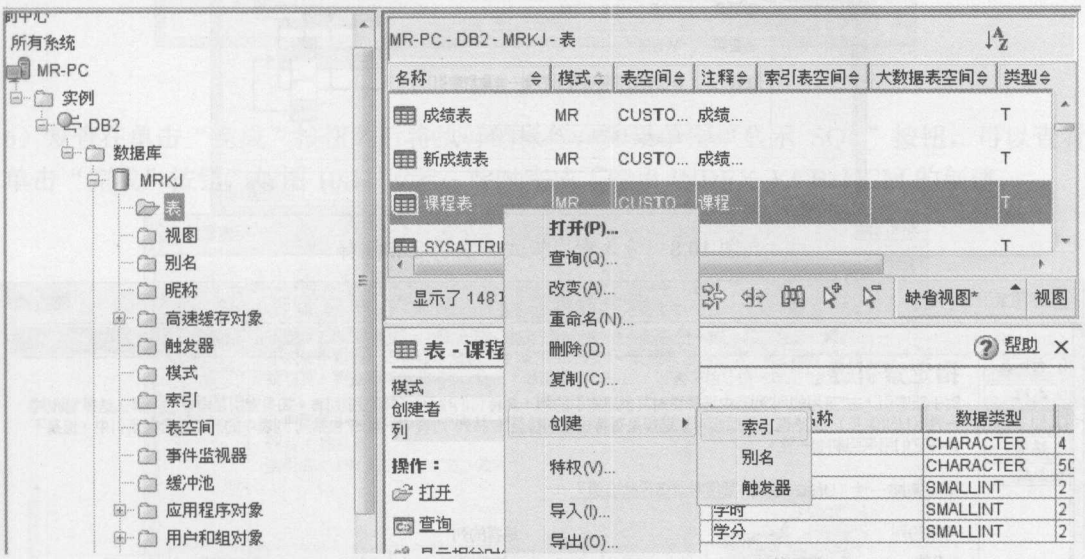


图 10.6　创建索引菜单

（2）在打开的“创建索引”对话框中，在“XML 列选项”区选择“否”选项。可以看出 DB2 已经自动生成了表模式和表名，并且不允许修改。单击“下一步”按钮，如图 10.7 所示。

（3）指定新索引的模式和名称。使用默认的索引模式“MR”，用户也可以选择其他模式。输入索引名“INDEX_KCB_KCM”，输入注释“课程表的课程名列上的唯一非集群索引”，单击“下一步”按钮，如图 10.8 所示。

（4）指定索引键。选中“强制唯一性”复选框，在“可用的列”列表中选择“课程名”，然后单击“>”按钮，这时“课程名”选项就添加到了“选择的列”列表中。选中“升序”单选按钮，单击“下

一步”按钮，如图 10.9 所示。

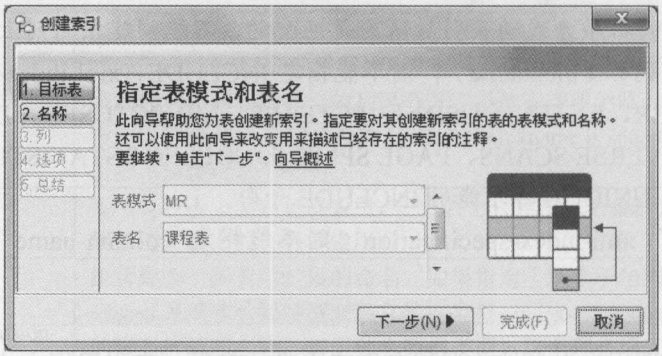


图 10.7　自动生成了表模式和表名

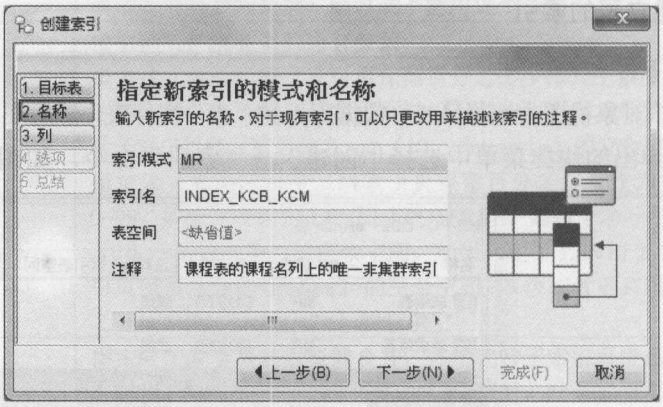


图 10.8　输入索引模式、索引名和注释

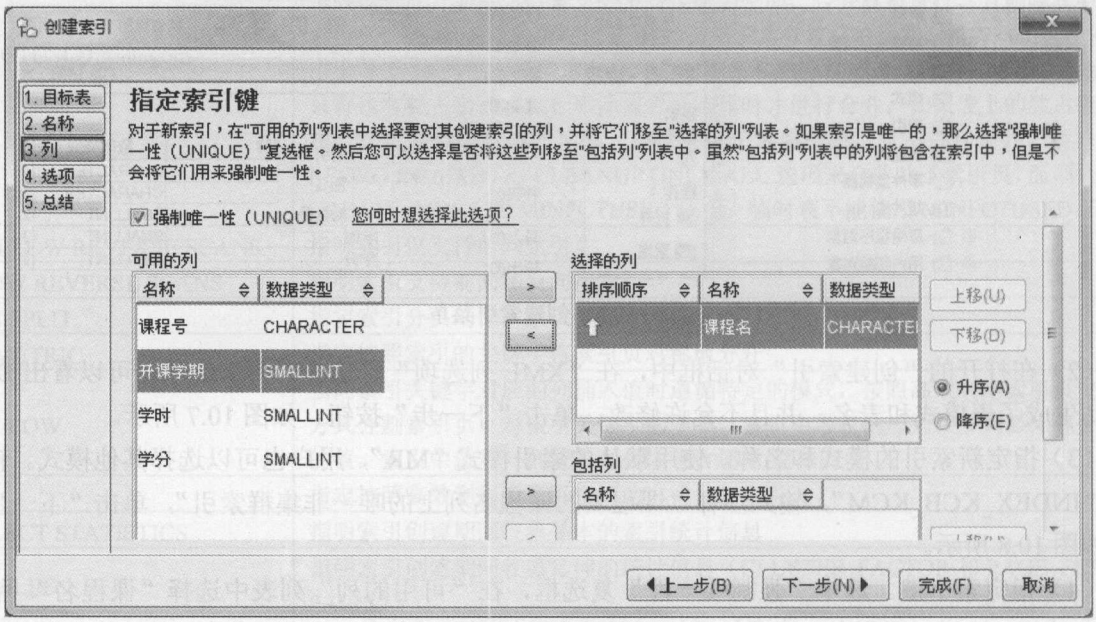


图 10.9　指定索引中的索引键

（5）指定索引选项。可以指定索引的详细特性，单击“下一步”按钮，接受默认选项，如图 10.10 所示。

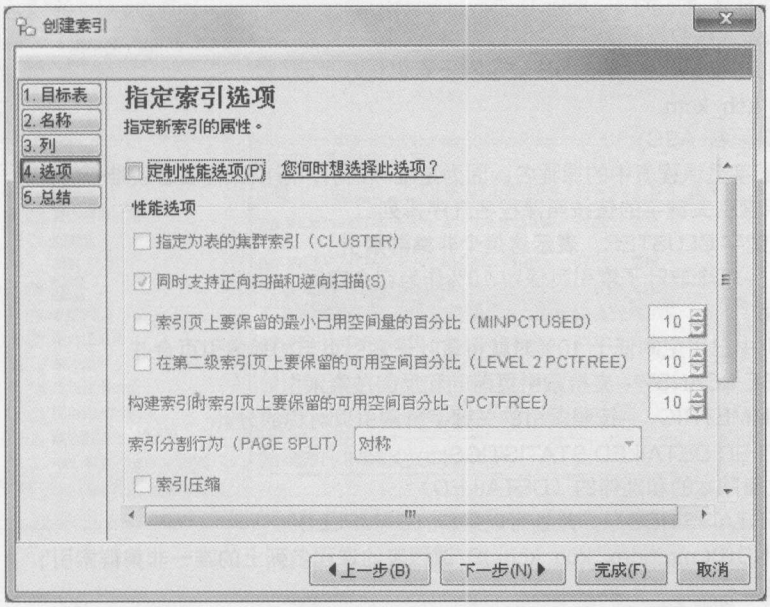


图 10.10　指定索引选项

（6）复查在单击“完成”按钮之后将执行的操作。如果单击“显示 SQL”按钮，可以查看 SQL 代码。单击“完成”按钮，如图 10.11 所示。此时完成了索引 INDEX_KCB_KCM 的创建。

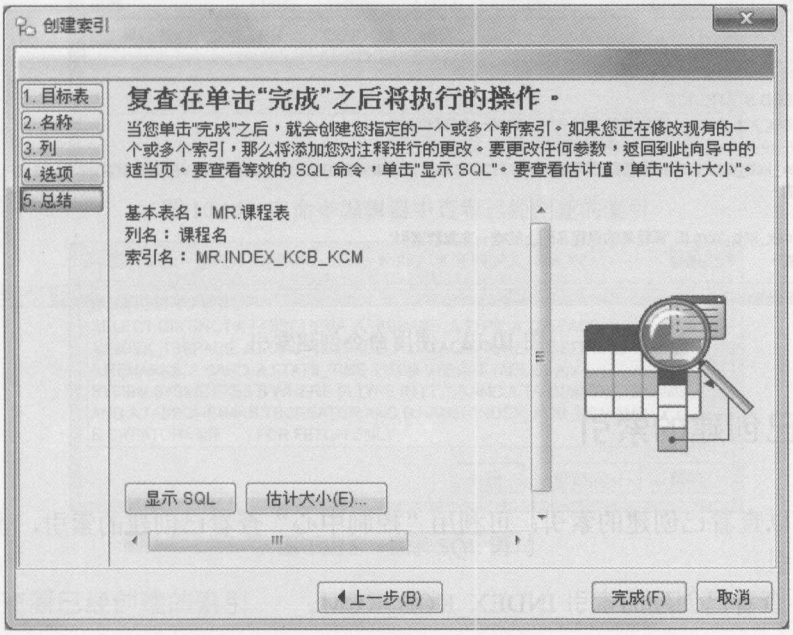


图 10.11　复查在单击完成之后将执行的操作

2. 通过命令建立表的索引

```
CONNECT TO mrkj USER mr USING 1;
--创建索引前首先要连接数据库
CREATE UNIQUE INDEX   --UNIQUE 表示唯一索引
        mr.index_kcb_kcm
ON mr.课程表 (课程名 ASC)
      --索引的关键字是课程表中的课程名，因为是唯一索引，所以课程名必须唯一
       --ASC 表示索引关键字的值按照课程名升序排列
       --没有用关键字 CLUSTER，表示这是个非集群索引
PCTFREE 10      --每个非叶子索引页保留 10%作为空余空间
MINPCTUSED 10
     --叶子索引页中已用空间低于 10%时就将该叶子索引页与相邻索引页合并
ALLOW REVERSE SCANS   --支持索引页前向和反向搜索索引
PAGE SPLIT SYMMETRIC   --按照索引的关键字将索引页对称的分开
COLLECT SAMPLED DETAILED STATISTICS;
--索引创建期间收集基本的和延伸的（DETAILED）
--索引统计信息（STATISTICS），并且可以采样（SAMPLED）
COMMENT ON INDEX mr.index_kcb_kcm IS '课程表的课程名列上的唯一非集群索引';
    --索引的注释
```

运行结果如图 10.12 所示。

```
---------------------------- 输入的命令 ----------------------------
create unique index  mr.index_kcb_kcm
on mr.课程表(课程名 ASC)
PCTFREE 10
MINPCTUSED 10
ALLOW REVERSE SCANS
PAGE SPLIT SYMMETRIC
COLLECT SAMPLED DETAILED STATISTICS;
COMMENT ON INDEX mr.index_kcb_kcm IS '课程表的课程名列上的唯一非集群索引';
------------------------------------------------------------------------
create unique index  mr.index_kcb_kcm on mr.课程表(课程名 ASC) PCTFREE 10 MINPCTUSED 10 ALLOW REVERSE SCANS PAGE SPLIT SYMMETRIC COLLEC
DB20000I  SQL 命令成功完成。

COMMENT ON INDEX mr.index_kcb_kcm IS '课程表的课程名列上的唯一非集群索引'
DB20000I  SQL 命令成功完成。
```

图 10.12　使用命令创建索引

10.2.4　查看已创建的索引

有两种方法可以查看已创建的索引：可利用“控制中心”查看已创建的索引，也可以利用 SQL 命令查看已创建的索引。

【例 10.7】　查看已创建的索引 INDEX_KCB_KCM。

1. 在“控制中心”窗口中查看已创建的索引

成功创建索引后，在“控制中心”对象视图中选择 MRKJ 数据库下的“索引”文件夹，右面内容窗

格中选择已创建的索引 INDEX_KCB_KCM，在该索引单击鼠标右键，在弹出的快捷菜单中选择“显示相关内容”命令，之后可以查看该索引的相关信息，如图 10.13、图 10.14 和图 10.15 所示。

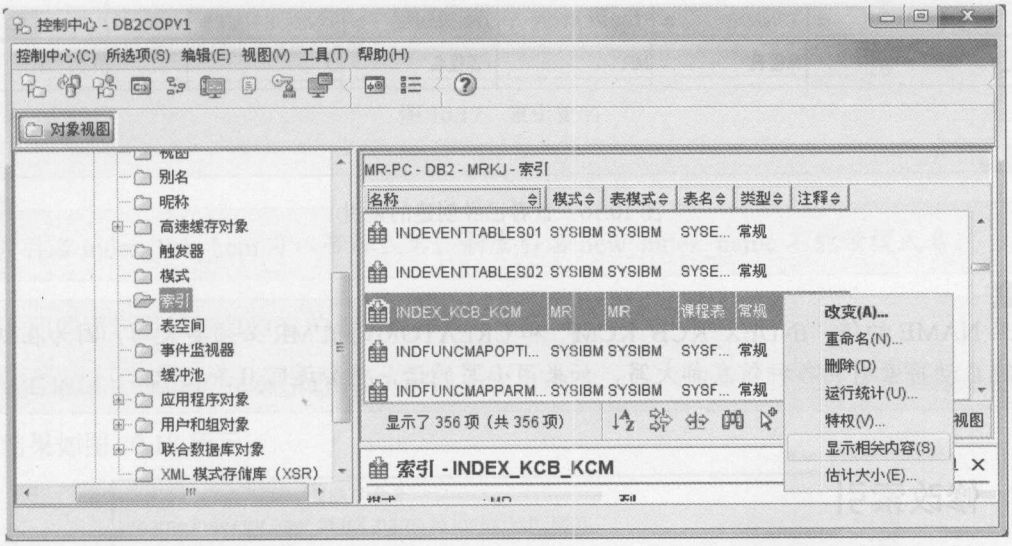


图 10.13　查看索引信息

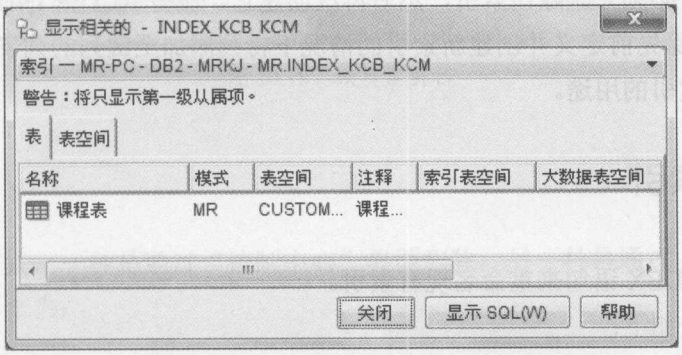


图 10.14　在命令编辑器中查看已经创建的索引

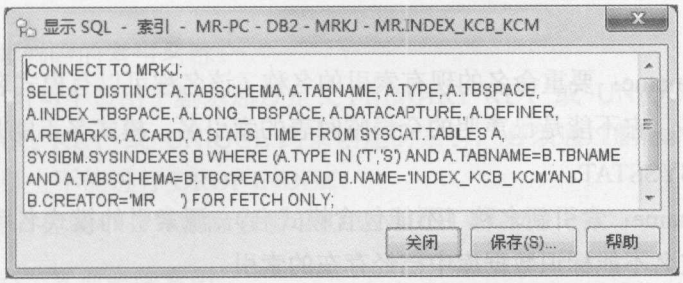


图 10.15　显示 SQL 语句

2. 使用命令查看已经创建的索引

在“命令编辑器”中，运行下面的命令也可以查询已经创建的索引 INDEX_KCB_KCM，如图 10.16 所示。

```
select * from sysibm.sysindexes
where name ='INDEX_KCB_KCM' AND CREATOR = 'MR';
```

NAME	CREATOR	TBNAME	TBCREATOR	COLNAMES	UNIQUERULE	COLCOUNT	IID
INDEX_KCB_K...	MR	课程表	MR	+课程名	U	1	

图 10.16　查看已经创建的索引

注意

这里 NAME 的值“INDEX_KCB_KCM”和 CREATOR 的值'MR'必须要大写，因为在创建索引时，系统自动将索引的各种信息都大写，如果用小写的话，将会返回 0 条记录。

10.2.5　修改索引

如果要修改索引，必须先删除该索引，然后再次创建它。没有 ALTER INDEX 语句。

例如，不能在未删除先前定义并创建新索引的情况下将列添加至键列的列表，可以使用 COMMENT 语句添加注释来描述索引的用途。

10.2.6　重命名索引

使用 RENAME INDEX 语句来重命名现有索引。

重命名索引的语法如下：

```
RENAME INDEX source_index_name TO target_index_name
```

参数说明如下。

- ☑ source_index_name：要重命名的现有索引的名称（该名称可以有模式名），必须标识数据库中已存在的索引。它不能是已声明的全局临时表的索引名。模式名不能是 SYSIBM、SYSCAT、SYAFUN 或 SYSSTAT。
- ☑ target_index_name：索引新名称（不能包含模式名）。源索引的模式名用于限定索引的新名称。新索引的限定名不能标识数据库中已经存在的索引。

【例 10.8】　将索引 mr.index_kcb_kcm 更名为 new_index_name（**实例位置：光盘\TM\sl\10\6**）。

```
RENAME INDEX mr.index_kcb_kcm TO new_index_name;
```

本例运行结果如图 10.17 所示。

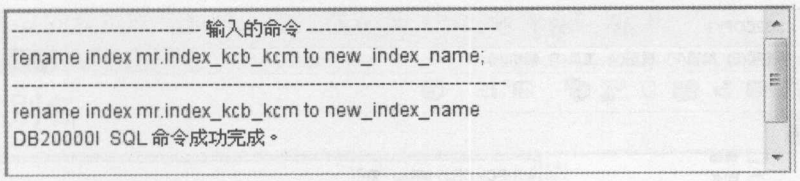


图 10.17　索引更名

注意

源索引名 index_kcb_kcm 可以带模式名，新索引名 new_index_name 不能带模式名。

使用下面的语句可以恢复原来的索引名：

```
RENAME INDEX mr.new_index_name TO index_kcb_kcm;
```

运行结果如图 10.18 所示。

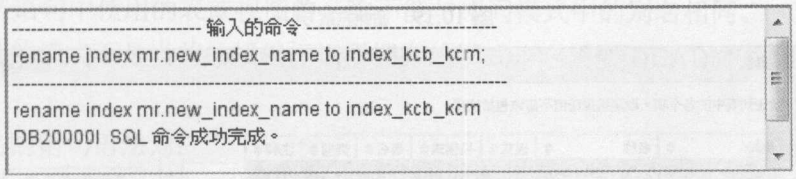


图 10.18　恢复索引名

10.2.7　删除索引

删除索引有两种方式，一种是通过“控制中心”界面删除，另一种是通过 SQL 语句删除。

删除索引的语句语法为：

```
DROP INDEX index_name;
```

参数说明如下。

☑ index_name：要删除的索引名称。

☑ DROP INDEX 语句不适用于删除通过定义 PRIMARY KEY 或 UNIQUE 约束创建的索引，若要删除 PRIMARY KEY 或 UNIQUE 约束创建的索引，必须通过删除约束实现。

☑ 在系统标的索引上不能进行 DROP INDEX。

【例 10.9】　通过“控制中心”删除索引 INDEX_KCB_KCM。

1. 通过“控制中心”界面删除索引

（1）在“控制中心”对象视图中找到数据库 MRKJ 下的“索引”，在内容窗格中找到索引 INDEX_KCB_KCM，然后单击鼠标右键，在弹出的快捷菜单中选择“删除”命令，如图 10.19 所示。

（2）在弹出的删除索引“确认”对话框中验证对话框中所列的项。验证无误后，单击“确定”按钮删除索引，如图 10.20 所示。

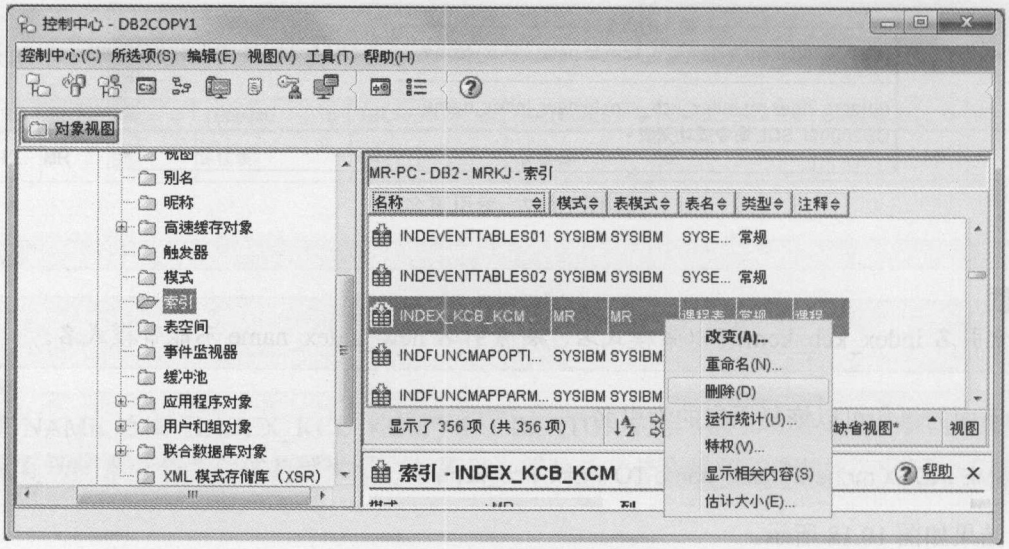


图 10.19　删除索引

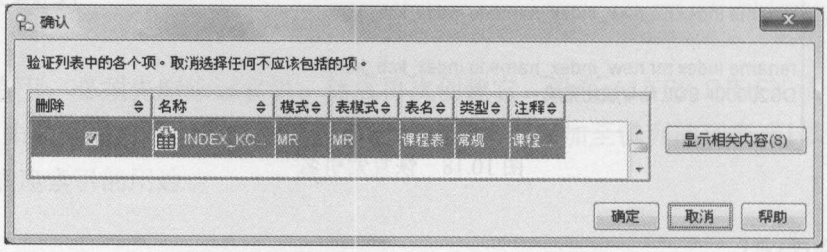


图 10.20　删除索引“确认”对话框

2. 通过命令删除索引

下面命令与上面的删除 MRKJ 数据库中的 INDEX_KCB_KCM 索引的操作等效。

```
DROP INDEX mr.index_kcb_kcm;
```

10.3　别　　名

别名是表或视图的另一个名称。如果可以引用现有表或视图，那么可以使用别名来引用该表或视图。

并不是所有上下文中都可以使用别名。例如，在检查约束的检查条件中不能使用别名。别名不能引用已声明的临时表。

与表或视图一样可以创建和删除别名，并且它可以具有关联的注释。但是，与表不同的是，别名可以在称为链接的过程中相互引用。别名是公共引用的名称，因此使用别名时不需要特殊权限或特权，但是，访问别名所应用的表或视图需要与这些对象关联的权限。

还有其他类型的别名，例如：数据库别名（也就是用 UNCATALOG/CATAOG 语句编目数据库）

和网络别名。还可以为表示联合系统上的数据表或视图的昵称创建别名。

10.3.1　创建别名

别名是引用表、昵称或视图的间接方法，这样 SQL 或 XQuery 语句可与该表或视图的限定名无关。只有当表名或视图名更改的情况下，才必须更改别名定义。别名可以在视图或触发器定义以及任何 SQL 或 XQuery 语句中使用，但表检查约束定义除外，在该定义中可以引用现有的表名或视图名。

可以为不存在的表、视图或别名定义别名。但是，当编译包含该别名的 SQL 或 XQuery 语句时，它必须存在。

别名可以在任何可使用限于表名的地方使用，且在别名链中不存在循环引用或重复引用的情况下，可以引用另一个别名。

别名不能与现有表、视图或别名同名，而只能引用同一个数据库中的一个表。在 CREATE TABLE 或 CREATE VIEW 语句中使用的表或视图的名称不能与相同模式中的别名相同。

除非别名所处的模式不是当前的授权标识所拥有的模式，它需要 DBADM 权限，否则，创建别名不需要特权。

创建表的别名的语句语法为：

```
CREATE ALIAS alias_name FOR table_name;
```

注意

DB 2 OS/390 版或 z/Series 版使用两种不同概念的别名：ALIAS 和 SYNONYM。这两种概念在 DB2 数据库中是有区别的。

（1）DB2 OS/390 版或 z/Series 版中的 ALIASe：

☑ 要求它们的创建者具有特殊的权限或特权。

☑ 不能引用其他别名。

（2）DB2 OS/390 版或 z/Series 版中的 SYNONYM：

☑ 只能被它们的创建者使用。

☑ 始终是非限定的。

☑ 删除引用的表时被删除。

☑ 不要与表或视图共享名称空间。

【例 10.10】　为“学生表”创建别名“学生信息表”（**实例位置：光盘\TM\sl\10\7**）。

```
CREATE ALIAS mr.学生信息表 FOR mr.学生表;
```

本例运行结果如图 10.21 所示。

学号	姓名	专业	性别	出生日期	总分数
14001	马芳	计算机应用技术 ...	女	1989-8-16	
14002	宋静	计算机应用技术 ...	女	1990-5-12	
14003	周天星	计算机应用技术 ...	男	1989-2-17	
14004	蒋阳泉	计算机应用技术 ...	男	1991-7-13	
14005	董敏	计算机应用技术 ...	女	1992-8-5	
14006	万方	计算机应用技术 ...	女	1993-8-12	
14007	秦冬雪	学前教育 ...	女	1989-5-6	
14008	欧阳云	学前教育 ...	男	1992-12-13	
14009	董湘芬	学前教育 ...	女	1989-8-16	
14010	禁传奇	学前教育 ...	男	1989-12-8	
14969	赵亮	计算机应用技术 ...	男	1989-8-30	

图 10.21　为“学生表”创建别名

10.3.2　使用别名

在编译语句时，别名被表名或视图名替换。如果别名或别名链不能被解析为表名或视图名，那么将导致错误。

如果“学生信息表”是“学生表”的一个别名，那么在编译时：

```
SELECT * FROM mr.学生信息表;
```

就会使下面的语句生效：

```
SELECT * FROM mr.学生表;
```

10.3.3　删除别名

删除别名时，将从目录中删除其描述，使引用该别名的任何程序包和高速缓存的动态查询失败，并且将从属于该别名的所有视图和触发器都标记为不可用。

删除别名的语句语法为：

```
DROP ALIAS alias_name;
```

【例 10.11】　删除“学生表”的别名“学生信息表”（**实例位置：光盘\TM\sl\10\8**）。

```
DROP ALIAS mr.学生信息表;
```

10.4　小　　结

本章介绍了如何在 DB2 数据库中对索引和别名进行一些操作，读者应该对默认值有一定的了解，

对于 DB2 的非空约束、唯一约束、检查约束、主键约束、外键约束要有深刻的理解，这些约束在其他程序语言的数据库应用程序开发中使用比较频繁。

10.5　实践与练习

（1）说明默认值的含义并举例说明。（**答案位置：光盘\TM\sl\10\9**）

（2）说说索引的概念、作用和分类。（**答案位置：光盘\TM\sl\10\10**）

（3）集群索引和非集群索引有何区别？为什么每个表只能有一个集群索引？（**答案位置：光盘\TM\sl\10\11**）

第11章

SQL过程语言

（视频讲解：120分钟）

本章主要介绍如何使用游标、流程控制语句与异常控制机制，主要包括游标的概念、声明游标、打开游标、读取游标、关闭游标以及游标和结果，并讲述条件语句、迭代语句与退出循环语句、RETURN语句、声明有名称的条件、声明条件处理程序和强制发出异常等内容。通过本章的学习，读者可以掌握游标的使用，掌握如何改变计算机的执行流程以满足应用程序的需要，并学会应用异常处理机制解决异常，同时掌握在数据的转换和装载过程中使用MERGE语句在一次操作期间对目标表与源表的匹配行进行更新和删除。

通过阅读本章，您可以：

- 了解SQL PL语言
- 掌握数据类型
- 掌握声明变量
- 熟悉赋值
- 掌握游标的使用方法
- 熟悉流程控制语句
- 熟悉异常处理机制
- 掌握MERGE语句

11.1　SQL PL 语言

DB2 SQL 过程语言（SQL Procedural Language，SQL PL）是 SQL Persistent Stored Module 语言标准的一个子集，该标准结合了 SQL 访问数据的方便性和编程语言的流控制。通过 SQL PL 当前的语句集合和语言特性，可以用 SQL PL 开发综合的、高级的程序，例如函数、存储过程和触发器，这样便可以将业务逻辑封装到易于维护的数据库对象中，从而提高数据库应用程序的性能。

SQL PL 支持本地和全局变量，包括声明和赋值，还支持条件语句和迭代语句、控制语句的转移、错误管理语句等，并提供返回结果集的方法。

11.1.1　SQL PL 语言元素

SQL PL 是一种用来定义和操纵数据库对象的语言。使用 SQL PL 可以定义数据库表、将数据插入表中、修改表中的数据和从表中检索数据。与所有语言一样，SQL PL 也定义了语法和一套语言元素。

SQL PL 语句包含以下一个或多个语言元素。

1．单字节的字符

SQL 的基本符号是单字节字符，可以是字母（A～Z、a～z、$）、#和@，或某个扩展字符集的成员、数字（0～9）或特殊字符。

其中，特殊字符是表 11.1 中的任何一个字符。

表 11.1　SQL 特殊字符

字　符	描　述	字　符	描　述
	空格	!	感叹号
.	句号	>	大于号
,	逗号	%	百分号
;	分号	/	斜杠
:	冒号	&	与
*	星号	(	左括号
‘	单引号	)	右括号
“	双引号	?	问号
-	减号	+	加号
_	下划线	|	竖直号
=	等于号	^	插字号
<	小于号		

2．权标

一个权标是一个或多个字符的序列。权标不可以包含空格，除非它是一个分界标识符或字符串常

量。权标可以是普通的权标（例如一个普通的标识符、一个数字常量或一个关键字），也可以是一个分界符权标（例如，一个分界标识符、一个字符串常量或一个运算符符号）。

任何权标都可包含小写字母，但普通权标中的小写字母通常被转换成大写字母，例如：

```
select * from mr.学生表;
```

转换成大写字母之后相当于：

```
SELECT * FROM MR.学生表;
```

3．标识符

SQL 标识符是用来形成一个名称的权标，标识符可以是一个普通标识符，后面跟有零个或多个字符的字母，每个字符都是一个大写字母、数字或带下划线的字符，例如 EMPLOYEE；标识符也可以是一个分界标识符，即用双引号括起来的一个或多个字符，例如“employee”。

4．值的数据类型

决定 DB2 如何解释这个值。DB2 支持许多内置的数据类型，还支持用户定义的类型（UDT）。

5．常量

常量指定一个值。它们分为字符、图形或十六进制字符串常量，以及整数、小数或浮点数字常量。

6．特殊寄存器

特殊寄存器是数据库管理程序为一个应用程序定义的一个存储区域，用来存储可以在 SQL 语句中引用的信息。特殊寄存器的示例是 CURRENT DATE、CURRENT DBPATRIRIONNUM 和 CURRENT SCHEMA。

7．例程

可以是函数、方法或过程。

表达式指定一个值。有字符串表达式、算术表达式和情况表达式，情况表达式可以用来根据对一个或多个条件的计算来指定某一结果。

8．谓词

指定关于给定的行或组的一个条件，结果可以是真、假或未知。

11.1.2 VALUES 语句

VALUES 语句的语法格式为：

```
VALUES {scalar_value | scalar_function} [,...n]
```

参数说明：

☑ scalar_value 是标量值。

☑ scalar_function 是标量函数（可以是系统标量函数或用户定义标量函数）。

☑ [,...n]表示可以同时输出多个标量值或标量函数，中间用逗号隔开，但这些标量值或标量函数的输出类型必须兼容（如都是数字型或都是字符型）。

☑ VALUES 语句的作用是：系统创建一个虚拟表，虚拟表只有一个列，列名为“1”，把每个标量值或标量函数的输出值都看成虚拟表中的一行记录，并在屏幕上或“命令编辑器”的“查询结果”选择项卡中显示出这个虚拟表。

【例 11.1】 在“命令编辑器”中运行下面命令可输出一列标量的值（**实例位置：光盘\TM\sl\11\1**）。

```
connect to mrkj user mr using 1;
VALUES 145,4.7,0.49
```

本例运行结果如图 11.1 所示。

1
145.00
4.70
0.49

图 11.1　查看标量值

11.2　数 据 类 型

一个值的数据类型决定了 DB2 如何解释该值，DB2 支持大量的内置数据类型，另外，用户还可以创建用户定义数据类型（UDT）来处理特殊情况。

DB2 支持的用户定义数据类型有：

☑ 单值数据类型。

☑ 结构数据类型。

☑ 数组数据类型。

11.2.1　系统数据类型

系统数据类型又称为内置数据类型，关于 DB2 系统数据类型的详细讲解请参见第 5 章 5.2 节。

11.2.2　创建单值数据类型

单值数据类型可看作是系统数据类型的别名，用户自定义数据类型可以在创建表、查询更改表、

转换数据类型、函数、存储过程等场合像系统数据类型一样使用。

在多表操作的情况下，当多个表中的列要存储相同类型的数据时，往往要确保这些列具有完全相同的数据类型、长度。

【例 11.2】 将 DEPT 表和 EMP 表中的共同字段 DEPTNO 定义成一个 sid_type 数据类型。

说明

从 MRKJ 数据库的 DEPT 表和 EMP 表的结构可以看出 DEPT 表中的 DEPTNO 字段与 EMP 表中的 DEPTNO 字段具有相同的数据类型，都为字符型的值，长度可定义为 2。

可以先定义这一数据类型，命名为“sid_type”，用于描述 DEPTNO 字段的这些属性。

用户自定义数据类型 sid_type 后，可以重新设计 MRKJ 数据库中 EMP 表和 DEPT 表中的 DEPTNO 字段，将 EMP 表中的 DEPTNO 字段和 DEPT 表中的 DEPTNO 字段定义为 sid_type 数据类型，如表 11.2、表 11.3 和表 11.4 所示。

表 11.2 自定义类型 sid_type

依赖的系统类型	值允许的长度
CHARATER	2

表 11.3 EMP 表中 DEPTNO 字段的重新设计

字 段 名	类 型
DEPTNO	sid_type

表 11.4 DEPT 表中 DEPTNO 字段的重新设计

字 段 名	类 型
DEPTNO	sid_type

通过上例可知：使用用户自定义类型前要先定义该类型，然后用该类型定义字段或变量。

创建用户自定义数据类型时首先应考虑如下两个属性：

（1）数据类型名称。

（2）新数据类型所依据的系统数据类型（又称为基类型）。

可通过界面和命令两种方法创建自定义数据类型。

1. 通过界面创建用户自定义数据类型

【例 11.3】 使用界面创建用户自定义的数据类型 sid_type。

实现步骤如下：

在“控制中心”对象视图中选择 MRKJ 数据库下的“应用程序对象”，然后选择“用户定义的单值类型”文件夹，如图 11.2 所示。在内容窗格中的空白处单击鼠标右键，在弹出的快捷菜单中选择“创建”命令，如图 11.3 所示，从而打开如图 11.4 所示的对话框。

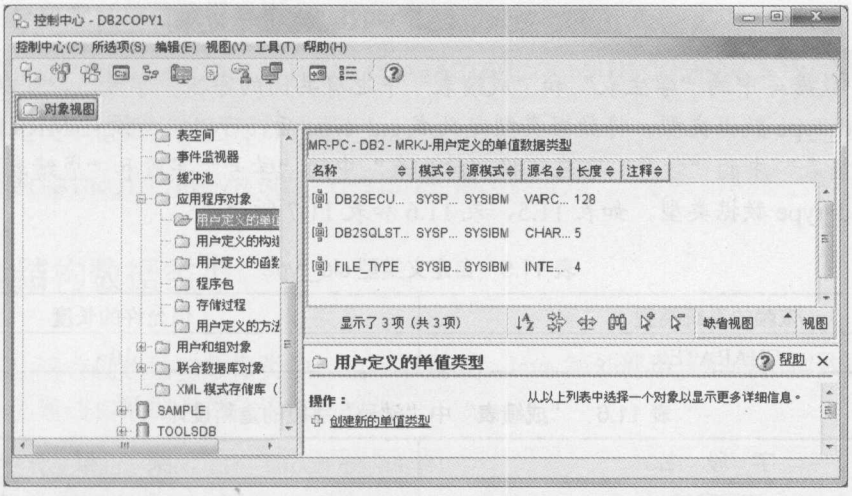


图 11.2　打开“用户定义的单值类型”

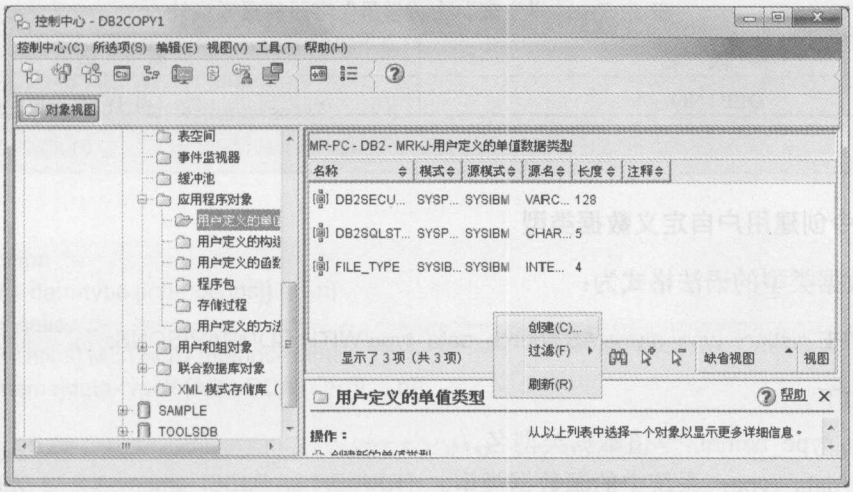


图 11.3　创建用户数据类型

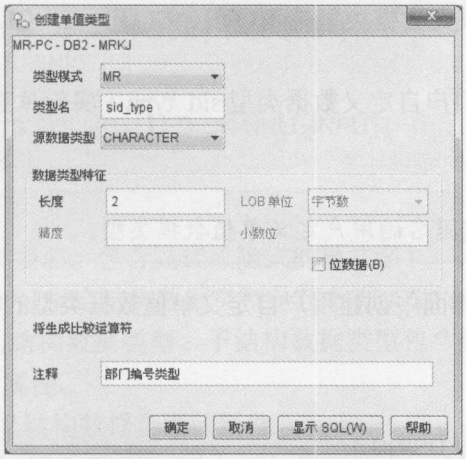


图 11.4　输入自定义单值数据类型的属性

说明

在 MRKJ 数据库中的“学生表”和“成绩表”中也有共同的字段“学号”，可以把这个共同的字段定义成 sid_type 数据类型。将数据类型定义成 sid_type 后，可以重新设计 MRKJ 数据库中“学生表”和“成绩表”中的“学号”字段，将“学生表”中的“学号”字段和“成绩表”中的“学号”字段定义为 sid_type 数据类型，如表 11.5、表 11.6 和表 11.7 所示。

表 11.5 自定义类型 sid_type

依赖的系统类型	值允许的长度
CHARATER	10

表 11.6 “成绩表”中“学号”字段的重新设计

字 段 名	类 型
DEPTNO	sid_type

表 11.7 “学生表”中“学号”字段的重新设计

字 段 名	类 型
DEPTNO	sid_type

2. 通过命令创建用户自定义数据类型

创建单值数据类型的语法格式为：

```
CREATE TYPE distinct_type_name AS source_data_type WITH COMPARISONS;
```

参数说明如下。

- ☑ distinct_type_name：单值数据类型名。
- ☑ source_data_type：系统中的源数据类型。
- ☑ WITH COMPARISONS：指定要创建系统生成的比较运算符，且该运算符用于比较一个单值类型的两个实例。如果元数据类型是 BLOB、CLOB、DBCLOB 或 VARCHAR，则不能用这些关键字。

【例 11.4】 用命令创建用户自定义数据类型 sid_type（**实例位置：光盘\TM\sl\11\2**）。

注意

不能在同一数据库中创建同名的用户定义单值数据类型。

下面的命令与例 11.3 中在界面中创建用户自定义单值数据类型的操作等效。

实现步骤如下。

（1）连接到 MRKJ 数据库：

```
CONNECT TO MRKJ USER mr USING 1;
```

（2）创建用户自定义单值数据类型 sid_type：

```
CREATE DISTINCT TYPE   MR.SID_TYPE AS CHARACTER (2)   WITH COMPARISONS ;
```

（3）为用户定义单值数据类型添加注释：

```
COMMENT ON DISTINCT TYPE MR.SID_TYPE IS '部门编号类型';
```

11.2.3　创建结构数据类型

可以使用 SQL 语言创建结构数据类型，也可以用 C、Java 等外部语言创建结构数据类型。用 SQL 命令创建结构数据类型的语法格式为：

```
CREATE TYPE type-name [UNDER   supertype-name]
     AS (attribute-definition)
[INSTANTIABLE | NOT INSTANTIABLE]
[INLINE LENGTH integer] [WITHOUT COMPARISONS][NOT FINAL]
MODE DB2SQL[WITH FUNCTION ACCESS][REF USING rep-type]
[CAST(SOURCE AS REF)WITH funcname1]
[CAST(REF AS SOURCE)WITH funcname2]
[method-specification]
```

其中：

```
attribute-definition ::=
{attribute-name data-type [lob-options]} [,...n]
method-specification ::=
     [OVERRIDING] METHOD method-name
[{[parameter-name]data-type2 [AS LOCATOR]}[,...n]]
RETURNS
     [data-type3[AS LOCATOR]|data-type4 CAST FROM data-type5[AS LOCATOR]]
     [SPECIFIC specific-name] [SELF AS RESULT]
     SQL-routine-characteristics
SQL-routine-characteristics ::=
[LANGUAGE SQL] [PARAMETER CCSID <ASCII | UNICODE>]
[NOT DETERMINISTIC | DETERMINISTIC]
[EXERNAL ACTION | NO EXTERNAL ACTION]
[READS SQL DATA | CONTAINS SQL] [CALLED ON NULL INPUT]
[INHERIT SPECIAL REGISTERS]
```

参数说明如下。

- ☑ type-name：结构数据类型名，包含显式或隐式的模式名。
- ☑ UNDER supertype-name：指定结构数据类型是超数据类型（supertype）下的子类型，超数据类型必须是已经存在的结构数据类型。子结构数据类型包含数据类型的所有属性，并且包含 attribute-definition 中的属性。
- ☑ attribute-definition：定义结构数据类型的属性。
- ☑ attribute-name：属性名。属性名不能与本结构数据类型和父结构数据类型中的任何属性名相同。

- ☑ data-type：属性的数据类型，它是 CREATE TABLE 语句中可以使用的那些数据类型（LONG VARCHAR、LONGVARGRAPHIC、XML 或基于 LONG VARCHAR 或 LONG VARGRAPHIC 的单值数据类型除外）。
- ☑ lob-options：指定大对象数据类型（LOB）或基于 LOB 类型的单值类型相关的选项。
- ☑ INSTANTIABLE | NOT INSTANTIABLE：指定类型是否可以创建实例。
- ☑ INLINE LENGTH integer：指定结构类型列实例的最大字节数。结构类型的默认的 INLINE LENGTH 由系统计算。
- ☑ WITHOUT COMPARISONS：指定结构类型的实例不支持被引用的函数。
- ☑ NOT FINAL：指定结构类型可能作为超数据类型。
- ☑ MODE DB2SQL：这一子句是必需的，允许直接调用这一结构数据类型上的构造函数。
- ☑ WITH FUNCTION ACCESS：指定这个类型和它的子类型的所有方法（包含以后创建的方法）可以用函数符号访问。这个子句只能用在类型层次的根结构类型中。
- ☑ REF USING rep-type：指定一个内置数据类型与这个结构类型和它的所有子类型中的某个引用类型是同义词。如果在结构类型层次的根类型中没有指定这个子句，那么系统就会指定“REF USING VARCHAR(16)FOR BIT DATA”。
- ☑ CAST(SOURCE AS REF)WITH funcname1：系统生成一个用于将 rep-type 数据类型的值转换成这个结构类型中的引用类型的函数，funcname1 是这个系统生成的函数的名称。
- ☑ CAST(REF AS SOURCE)WITH funcname2：系统生成一个用于将这个结构类型中的引用类型的值转换成 rep-type 数据类型的函数，funcname2 是这个系统生成的函数的名称。
- ☑ method-specification：定义这个结构类型的方法。直到用 CREATE METHOD 语句创建一个方法后，这个方法才能被实际使用。
- ☑ OVERRIDING：指定这个方法重写这个结构类型的超结构类型的方法。
- ☑ method-name：方法名。
- ☑ parameter-name：参数名，不能是 SELF，SELF 是方法的隐式目标参数。如果是 SQL 方法，则必须为每个参数命名。如果方法被申明为重写另一个方法，参数名必须与被重写的方法的相关参数名相同，否则会返回错误。
- ☑ data-type2：指定每个参数的数据类型，这些参数数据类型是 CREATE TABLE 语句中可用的参数类型。最多允许 90 个参数，包含隐式 SELF 参数。
- ☑ AS LOCATOR：对于 LOB 类型或基于 LOB 类型的单值类型，可以添加 AS LOCATOR 子句，它指定 LOB 定位器传递到方法，而不是实际值传递到方法。
- ☑ RETURNS：这个必需的子句指定参数的返回值。
- ☑ data-type4 CAST FROM data-type5：指定方法返回值的数据类型。data-type5 必须能够转换到 data-type4 参数，如果不可转换，则返回错误。
- ☑ SPECIFIC specific-name：提供被定义的方法实例的唯一的特定名称。
- ☑ SELF AS RESULT：指定方法是保留类型的方法，意思是指被声明的返回必须与被声明的目标类型相同；当一个 SQL 语句被编译和分解到一个类型保留的方法，则方法结果的静态类型与目标参数的静态类型相同；方法必须被这样实现，即结果的动态类型必须与目标参数的动态类型相同，且结果不能为空。如果方法被定义为重写另一个方法，则不能指定这个子句。
- ☑ SQL-routine-characteristics：指定这个结构类型中方法体的特性。将要用 CREATE METHOD

语句创建方法。

- ☑ LANGUAGE SQL：指定方法是包含一个 RETURN 的 SQL 方法。
- ☑ PARAMETER CCSID：指定传参编码方式，对 UNICODE 数据库默认值为 PARAMETER CCSID UNICODE，对其他数据库默认值是 PARAMETER CCSID ASCII。
- ☑ NOT DETERMINISTIC | DETERMINISTIC：指定给定参数值后，方法总是返回相同的结果（DETERMINISTIC），或者方法依赖于影响结果的一些状态值（NOT DETERMINISTIC）。
- ☑ EXTERNAL ACTION | NO EXTERNAL ACTION：指定方法是否采取措施改变非数据库管理器托管对象的一些状态。如果指定了 EXTERNAL ACTION，则系统就会进行一些优化，假定方法没有外部影响的系统优化措施，就会被阻止，例如发送消息、发出鸣声、写文件记录等。
- ☑ READS SQL DATA：指定不修改 SQL 数据的 SQL 语句可以被方法执行。
- ☑ CONTAINS SQL：指定既不读取也不修改 SQL 数据的 SQL 语句可以被方法执行。
- ☑ CALLED ON NULL INPUT：指定调用用户定义的方法时，忽略任意参数是否为空。如果方法被定义为重写另一个方法，则不能指定这个子句。NULL CALL 可看成是 CALLED ON NULL INPUT 的同义词。
- ☑ INHERIT SPECIAL REGISTERS：继承专用寄存器。

【例 11.5】　创建一个数据类型层次，包含“学生结构类型”类型和“学生管理结构类型”子类型（**实例位置：光盘\TM\sl\11\3**）。

实现步骤如下。

（1）连接 MRKJ 数据库：

```
CONNECT TO mrkj USER mr USING 1;
```

（2）创建用户自定义结构数据类型：

```
CREATE TYPE mr.学生结构类型 AS
(
    学号 CHAR(6),
    姓名 CHAR(16),
    性别 CHAR(4)
)MODE DB2SQL;
```

（3）创建继承了“学生结构类型”的用户自定义结构数据类型“学生管理结构类型”：

```
CREATE TYPE mr.学生管理结构类型 UINDER 学生结构类型 AS
(
    专业 CHAR(40),
    总学分 SMALLINT,
    备注 VARCHAR(400)
)MODE DB2SQL;
```

说明

上面创建的两个用户自定义结构数据类型中，“学生结构类型”是超类型，“学生管理结构类型”是继承了“学生结构类型”的子类型。所以“学生管理结构类型”也继承了“学生结构类型”中的字段“学号”、“姓名”和“性别”。读者可以仿照例 11.4 将 EMP 表和 DEPT 表也创建一个数据类型层次，然后再将其删除。

使用下面的语句可以删除上面创建的两个结构类型：

```
DROP TYPE mr.学生管理结构类型;
DROP TYPE mr.学生结构类型;
```



注意

两个 DROP 语句的顺序不能颠倒，因为“学生管理结构类型”和“学生结构类型”之间有继承关系。先要删除子类型“学生管理结构类型”，然后再删除超类型“学生结构类型”。

11.2.4 创建数组数据类型

SQL 存储过程从 DB2 的 9.5 版本开始支持数组类型的变量和参数，要定义一个数组类型的变量，需要先在数据库中创建该类型，然后在过程或应用程序中声明它。数组是临时的值，可以在存储过程和应用程序中操纵它，但是不能将它存储到表中。

创建数组数据类型的语法为：

```
CREATE TYPE array_type_name AS datatype ARRAY [2147483647 | integer_constant]
```

参数说明：

- ☑ 数组类型的名称应该用模式加以限定，并且在当前服务器上应该是唯一的。
- ☑ LONG VARCHAR、LONG VARGRPAHIC、XML 和用户定义类型不能作为数组元素的数据类型。
- ☑ 整数 integer_constant 指定数组的最大基数，它是可选的。
- ☑ 数组元素可以通过 array_type_name [subindex]来引用，其中 subindex 必须介于 1 到数组的基数（2147483647 或 integer_constant）之间。
- ☑ DB2 支持一些操作数组的方法。例如，函数 CARDINALITY（array_type_name）返回一个数组中元素的个数。

【例 11.6】 在 DB2 中创建数组类型（**实例位置：光盘\TM\sl\11\4**）。

```
CREATE TYPE numbers as INTEGER ARRAY[10];
CREATE TYPE names as VARCHAR(15) ARRAY[];
CREATE TYPE MYSCHEMA.totalcomp as DECIMAL(16,2) ARRAY[];
```

11.3 声 明 变 量

SQL 过程允许使用本地变量赋予 SQL 值和获取 SQL 值，以支持所有 SQL 逻辑。在 SQL 过程中，使用本地变量之前要先进行声明。

变量的声明语法为：

```
DECLARE SQL_variable_name datatype
    [DEFAULT NULL | DEFAULT constant]
```

参数说明如下。

☑ SQL_variable_name：定义本地变量的名称。该名称不能与其他变量或参数名称相同，也不能与列名相同。

☑ DEFAULT 值：如果没有指定，在声明时将赋值为 NULL。

注意

从 DB2 V9.5 开始支持在一个 DECLARE 语句中声明多个相同数据类型的变量。

【例 11.7】　在 DB2 中声明变量（**实例位置：光盘\TM\sl\11\5**）。

```
DECLARE v_salary DEC(12,2) DEFAULT 0.0;
DECLARE v_status char(6) DEFAULT'YES';
DECLARE v1,v2 INT DEFAULT 0;
```

注意

上述几个例子不能独立运行，必须放在函数、存储过程或触发器中作为函数、存储过程或触发器的子句。

【例 11.8】　在 DB2 中创建一个存储过程（**实例位置：光盘\TM\sl\11\6**）。

实现步骤如下。

（1）连接到 MRKJ 数据库：

```
CONNECT TO mrkj USER mr USING 1#
```

（2）创建一个使用数组数据类型 numbers 和 names 的存储过程：

```
CREATE PROCEDURE mr.proc_varray_test
(OUT v_pnumb mr.numbers,OUT mynames mr.names)
BEGIN
DECLARE v_pnumb mr.numbers;
SET v_pnumb = ARRAY[1,2,3,5,7,11];
SET mynames[1]='MARINA';
END #
```

（3）调用存储过程 mr.proc_varray_test：

```
CALL mr.proc_varray_test(?,?)#
```

```
返回结果如下：

输出参数的值
------------------------
参数名：  V_PNUMB
参数值：  -


参数名：  MYNAMES
```

```
参数值： [MARINA]

返回状态 =0
```

11.4 赋　　值

在 DB2 中时常会用到赋值操作，本节将对如何为变量赋值进行详细讲解。

11.4.1 赋值语句的语法

为变量赋值的方法如下：

（1）SET 语句。

（2）VALUES INTO 语句。

（3）SELECT INTO 语句。

（4）FETCH INTO 语句。

简化后的 SET 语句的语法为：

```
SET variable_name = <value | expression | NULL>
```



说明

变量名 variable_name 可以是一个本地变量、全局变量或数组元素的名称。

【例 11.9】　SET 语句的用法（**实例位置：光盘\TM\sl\11\7**）。

```
SET num1=18;
SET total=(select sum(c1)from T1);
SET var1=POSSTR('MR', 'BEST');
SET v_numb = array[1,2,3,4];
```

【例 11.10】　VALUE INTO 语句的用法（**实例位置：光盘\TM\sl\11\8**）。

```
VALUES 2 INTO f5;
VALUES 'BEST' INTO v1;
```

【例 11.11】　SELECT INTO 语句的用法（**实例位置：光盘\TM\sl\11\9**）。

```
SELECT SUM (c1) INTO var1 FROM T1;
SELECT POSSTR ('MR', 'BEST') INTO v1 FROM SYSIBM.SYSDUMMY1;
```

11.4.2 专用寄存器

专用寄存器是一个存储块，可以供一个应用程序过程使用。寄存器中的值可以在 SQL 语句或 SQL

PL 语句中访问和引用。

最常用的专用寄存器如下：

（1）CURRENT DATE。

（2）CURRENT TIME。

（3）CURRENT TIMESTAMP。

（4）CURRENT USER。

（5）CURRENT PATH。

所有这些寄存器都可以通过在名称中加“下画线”来引用，例如：CURRENT_DATE。

有些专用寄存器的值可以通过 SET 语句来更新。例如：

（1）为了更新正在访问的模式，需要更改专用寄存器 CURRENT SCHEMA。

（2）若要更改默认函数路径，则需要更新专用寄存器 CURRENT PATH。

【例 11.12】　查询当前日期和时间（**实例位置：光盘\TM\sl\11\10**）。

在“命令编辑器”中运行下面两条语句：

```
VALUES CURRENT DATE;
VALUES CURRENT TIME;
```

本例运行结果如图 11.5 所示。

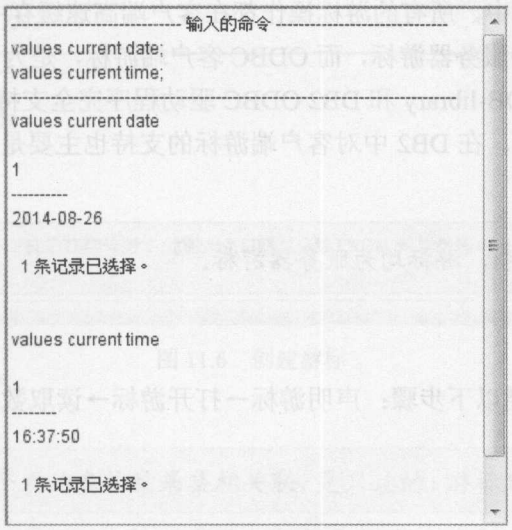


图 11.5　查询当前日期

【例 11.13】　更改专用寄存器 CURRENT SCHEMA（**实例位置：光盘\TM\sl\11\11**）。

```
SET CURRENT_SCHEMA=mr;
```

11.5　游　　标

游标提供了一种从表中检索数据并进行操作的灵活手段，游标主要用在服务器上，处理由客户端

发送给服务器端的 SQL 语句，或是批处理、存储过程、触发器中的数据处理请求。

11.5.1 游标的概念

1. 游标的概念和用途

游标实际上是一种能从包括多条数据记录的结果集中每次提取一条记录的程序机制。相当于通过程序对文件进行读写处理的文件句柄，能像文件句柄读取文件一样读取来自基础表的结果集，从而把表中数据以平面文件的形式提交给应用程序。

DB2 通过游标提供了对一个结果集进行逐行处理的能力，游标可看作是一种特殊的指针，它与某个查询结果相联系，可以指向结果集的任意位置，以便对指定位置的数据进行处理。使用游标可以在查询数据的同时对数据进行处理。

2. 游标的类型

在 DB2 中，有两类游标可以用于应用程序：前端（客户端）游标和后端（服务器端）游标。服务器端游标是由数据库服务器创建和管理的游标，而客户端游标是由 ODBC 和 DB-Library 支持、在客户端实现的游标。在客户端游标中，所有的游标操作都在客户端高速缓存中执行。最初实现 DB-Library 客户端游标时，DB2 尚不支持服务器游标，而 ODBC 客户端游标，是为了用于仅支持游标特性默认设置的 ODBC 驱动程序。由于 DB-library 和 DB2 ODBC 驱动程序完全支持通过服务器游标的游标操作，所以应尽量不使用客户端游标。在 DB2 中对客户端游标的支持也主要是考虑向后兼容。



说明

本节中如果没有特殊说明，游标均为服务器游标。

3. 使用游标的步骤

DB2 对游标的使用要遵循以下步骤：声明游标→打开游标→读取数据→关闭游标→删除游标。

11.5.2 声明游标

SQL PL 提供 DECLARE CURSOR 语句来定义一个游标，并提供其他语句来支持返回结果集和游标处理。

声明游标的语法如下：

```
DECLARE cursor_name CURSOR FOR
    select_statement
    [WITHOUT HOLD | WITH HOLD]
    [WITHOUT RETURN | WITH RETURN [TO CALLER | TO CLIENT]]
    [FOR READ ONLY | FOR UPDATE [OF column_name [,…n]]]
```

参数说明如下。

- ☑ select_statement 是一条有效的 SQL SELECT 语句。可以指定 FOR UPDATE 子句，以便将游标用于定位更新或删除。
- ☑ WITHOUT HOLD/WITH HOLD 选项定义 COMMIT 操作之后的游标状态（open/close）。默认情况下为 WITHOUT HOLD。如果使用了 WITH HOLD 选项定义一个游标，那么在 COMMIT 操作之后，该游标保持 OPEN 状态。在 ROLLBACK 操作之后，所有游标都将被关闭。
- ☑ READ ONLY：说明所声明的游标为只读的。
- ☑ FOR UPDATE：指出游标中可以更新的列，若有参数“OF column_name[,...n]”，则只能修改给出的这些列，若在 FOR UPDATE 中未指出列，则可以修改所有列。

【例 11.14】　声明一个只读游标，用于返回“计算机应用技术”专业的学生的“学号”“姓名”“性别”“出生日期”“总分数”（**实例位置：光盘\TM\sl\11\12**）。

```
DECLARE mr.xsb_cur1 CURSOR FOR
    SELECT 学号,姓名,性别,出生日期,总分数
    FROM mr.学生表
WHERE 专业='计算机应用技术'
FOR READ ONLY;
```

本例运行结果如图 11.6 所示。

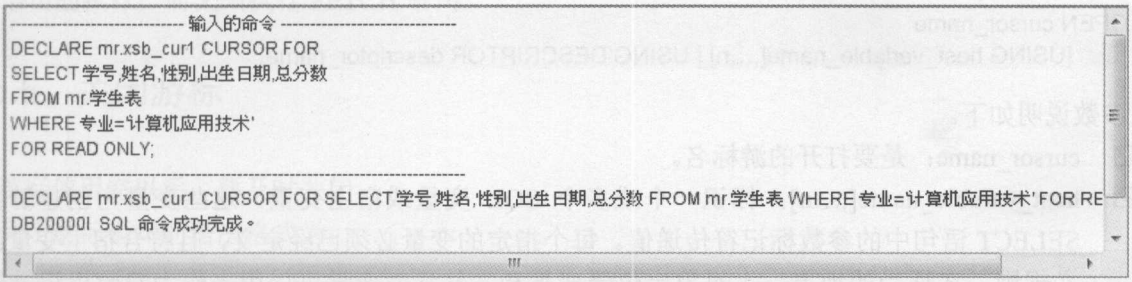


图 11.6　创建游标

说明

该语句定义的游标与单个表的查询结果集相关联，是只读的，游标只能从头到尾顺序提取数据。

【例 11.15】　声明一个游标，用于更新“计算机应用技术”专业的学生的“总分数”（**实例位置：光盘\TM\sl\11\13**）。

```
DECLARE mr.xsb_cur2 CURSOR FOR
    SELECT 学号,姓名,总分数
    FROM mr.学生表
WHERE 专业='计算机应用技术'
FOR UPDATE OF 总分数;
```

本例运行结果如图 11.7 所示。

```
---------- 输入的命令 ----------
DECLARE mr.xsb_cur2 CURSOR FOR
SELECT 学号,姓名,总分数
FROM mr.学生表
WHERE 专业='计算机应用技术'
FOR UPDATE OF 总分数;
---------- ----------
DECLARE mr.xsb_cur2 CURSOR FOR SELECT 学号,姓名,总分数 FROM mr.学生表 WHERE 专业='计算机应用技术' FOR UPDATE OF 总分
DB20000I  SQL 命令成功完成。
```

图 11.7　声明一个用于更新的游标

说明

该语句声明一个名为“XSB_CUR2”的游标，可对“总分数”列进行修改。

11.5.3　打开游标

声明游标后，要使用游标从中提取数据，就必须先打开游标。在 DB2 SQL 中，使用 OPEN 语句打开游标，语法格式为：

```
OPEN cursor_name
    [USING host_variable_name[,...n] | USING DESCRIPTOR descriptor_name]
```

参数说明如下。

- ☑ cursor_name：是要打开的游标名。
- ☑ host_varable_name[,...n]：标识一个或多个 SQL 变量或宿主变量。这些变量为创建游标的 SELECT 语句中的参数标记符传递值。每个指定的变量必须已经定义，且符合宿主变量的定义规则。在适当的地方，大对象定位器变量和文件参考变量可以用来作为参数标记符值的来源。
- ☑ DESCRIPTOR descriptor_name：指定一个 SQLDA 数据结构变量。它包含创建游标的 SELECT 语句所使用的宿主变量的合法描述信息。

OPEN 语句打开游标，然后通过执行在 DECLARE CURSOR（或 set cursor_variable）语句中指定的 DB2 SQL 语句填充游标（即生成与游标相关联的结果集）。

【例 11.16】　使用命令打开游标（**实例位置：光盘\TM\sl\11\14**）。

```
OPEN mr.xsb_cur1;
```

说明

游标被打开后，就可以提取其中的数据。

本例运行结果如图 11.8 所示。

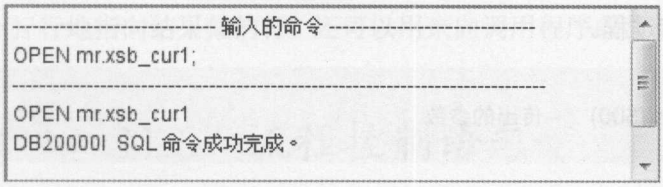


图 11.8　打开游标

11.5.4　读取游标

游标打开后，就可以使用 FETCH 语句从中读取数据。

FETCH INTO 语句的语法格式为：

```
FETCH[FROM ] cursor_name
[INTO host_varable_name [,...n] | USING DESCRIPTOR descriptor_name]
```

参数说明如下。

☑ host_variable_name[,...n]：宿主变量名列表，使用该列表中的项替换 SQL 语句中的占位符。

需要注意的是，如果 SQL 语句不是 SELECT 语句，那么在执行了游标的 OPEN 命令后，即完成了 SQL 语句的执行，而无须执行 FETCH 语句。

11.5.5　关闭游标

游标使用完以后，要及时关闭。

关闭游标语句的语法格式为：

```
CLOSE cursor_name[WITH RELEASE]
```

参数说明如下。

☑ WITH RELEASE：如果指定了 WITH RELEASE 选项，那么当游标关闭时，由读稳定（RS）或者可重复读（RR）隔离级别提供的一些保证就被终止了。特别地，如果使用这个选项关闭一个游标，然后重新打开，那么它就获得了新锁。

当一个事务终止时，如果声明的游标没有 WITH HOLD 选项，则该游标会被隐式关闭。

【例 11.17】　关闭游标（**实例位置：光盘\TM\sl\11\15**）。

```
CLOSE mr.xsb_cur1;
```

使用游标完整的步骤如下：

（1）定义一个数据表，表名为 test，然后插入 3 条记录。

```
create table test(id int , city char(20))
insert into test values (1,'wuhan'),(2,'hangzhou'),(3,'chengdu')
```

（2）定义一个存储过程。

```
create procedure est(
out v_message varchar(500)    --传出的参数
)
```

（3）定义变量，声明游标，读取游标。

```
BEGIN
declare v_city varchar(20);        --定义一个变量
declare v_cur cursor for select city from test for read only;   --声明游标
open   v_cur;                      --打开游标
fetch   v_cur into v_city;         --读取游标
set   v_message =v_city;           --为参数设置值
END #
```

（4）调用存储过程。

```
call est(?)#
```

运行结果如图 11.9 所示。

```
------------------------------ 输入的命令 ------------------------------
call est(?)#
------------------------------------------------------------------------
call est(?)

输出参数的值
------------------------
参数名：V_MESSAGE
参数值：-

返回状态 = 0
```

图 11.9　使用游标的完整步骤

11.5.6　游标和结果集

游标可以用于将结果集返回给调用程序或其他存储过程。

（1）WITHOUT RETURN/WITH RETURN 选项指定游标的结果集是否用于作为从一个过程中返回的结果集。

（2）WITH RETURN TO CALLER 选项指定将来自游标的结果集返回给调用者，后者可以是另一个过程或一个客户机应用程序。这是默认选项。

（3）WITH RETURN TO CLIENT 选项指定将来自游标的结果集返回给客户机应用程序，绕过任何中间的嵌套过程。

通过下面的方法可以从一个存储过程中返回结果集：

（1）创建一个存储过程，创建时指定 DYNAMIC RESULT SETS 子句。

（2）游标处理。

通过下面的方法，可以在一个存储过程中处理一个游标的结果：

（1）在存储过程块的开头部分声明（DECLARE）游标，声明时指定 WITH RETURN 子句。

（2）打开该游标。

（3）将游标的结果去除到之前已声明的本地变量中（隐式游标处理除外，在下面的 FOR 语句中将对此加以解释）。

（4）关闭该游标（注意：如果现在不关闭游标，当过程终止时将隐式地关闭游标）。在存储过程

中，游标不仅可以用来一行行地指向结果集的行，还可以用来向调用程序或其他存储过程返回结果集。

11.6　流程控制语句

设计程序时，常常需要利用各种流程控制语句，改变计算机的执行流程，以满足程序设计的需要。在 DB2 中提供了如表 11.8 所示的流程控制语句。

表 11.8　SQL PL 流程控制语句

控 制 语 句	说　明
IF 和 CASE	条件语句
LOOP、WHILE、REPEAT、FOR	迭代语句
ITERATE	用于重新开始下一次循环
LEAVE	用于退出其所在这一层的循环
RETURN	无条件返回

11.6.1　条件语句

条件语句用于根据某个表达式的值从若干条给定语句中选择一个来执行，主要包括 IF 语句和 CASE 语句两种，下面分别进行讲解。

1. IF 语句

通过 IF 语句可以根据一个条件的状态来实现逻辑的分支，对给定的条件进行判定，当条件为真或假时分别执行不同的 SQL PL 语句。IF 语句支持使用可选的 ELSE IF 子句和默认的 ELSE 子句。END IF 子句是必需的，它用于表明 IF 语句的结束。

IF 语句的语法格式为：

```
IF Boolean_expression THEN                          --条件表达式
    <sql_statement | statement_block>               --条件表达式为真时执行
    [ELSE <sql_statement | statement_block>]         --条件表达式为假时执行
END IF
```

参数说明如下。

- ☑ Boolean_expression：条件表达式，如果条件表达式中含有 SELECT 语句，必须用圆括号将 SELECT 语句括起来，运算结果为 TRUE（真）或 FALSE（假）。
- ☑ <sql_statement | statement_block>：DB2 SQL 语句或 BEGIN…END 定义的语句块。当 Boolean_expression 条件表达式的值为真或假要执行多条 DB2 SQL 语句时，这些语句要用在 BEGIN…END 之间，构成一个语句块。

可在 IF 区或 ELSE 区嵌套另一个 IF 语句，对于嵌套层数没有限制。

由上述语法格式，可看出条件语句分为带 ELSE 部分和不带 ELSE 部分两种使用形式。

（1）IF 条件表达式 THEN

```
A  --SQL PL 语句或语句块
ELSE
B  --SQL PL 语句或语句块
```

当条件表达式的值为真时，执行 A 然后执行 IF 语句的下一语句；条件表达式的值为假时执行 B，然后执行 IF 语句的下一语句。

（2）IF 条件表达式

```
A  --SQL PL 语句或语句块
```

当条件表达式的值为真时执行 A，然后执行 IF 语句的下一条语句；条件表达式的值为假时直接执行 IF 语句的下一条语句。

如果在 IF…ELSE 语句的 IF 区和 ELSE 区都使用了 CREATE TABLE 语句或 SELECT INTO 语句，那么，CREATE TABLE 语句或 SELECT INTO 语句必须使用相同的表名。

IF…ELSE 语句可用在批处理、存储过程（经常使用这种结构测试是否存在着某个参数）及特殊查询中。

【例 11.18】 如果“计算机应用技术”的课程的平均成绩大于等于 73 分，则显示“平均成绩大于等于 73 分”，否则显示“平均成绩低于 73 分”，根据此例编写一个包含 IF…ELSE 的非嵌套存储过程（**实例位置：光盘\TM\sl\11\16**）。

实现步骤如下。

（1）连接 MRKJ 数据库：

```
CONNECT TO mrkj USER mr USING 1#
```

（2）创建具有题目中逻辑的存储过程：

```
GREATE PROCEDURE mr.proc_ifclause(OUT text1 char(40))
BEGIN
    IF((select avg (成绩)FROM mr.学生表 x,mr.课程表 k,mr.成绩表 c
WHERE x.学号=c.学号 AND c.课程号=k.课程号 AND k.课程名='计算机应用技术')<73)
THEN SET text1='平均成绩低于 73';
ELSE SET text1='平均成绩大于等于 73';
END IF;
END #
```

（3）调用上面的存储过程：

```
CALL mr.proc_ifclause (?)#
```

【例 11.19】 编写一个包含嵌套的 IF…ELSE 语句的存储过程：如果“计算机应用技术”课程的平均成绩大于等于 73 分，则显示“平均成绩大于等于 73 分”，否则显示“平均成绩低于 73 分”（**实例位置：光盘\TM\sl\11\17**）。

实现步骤如下。

（1）连接 MRKJ 数据库：

```
CONNECT TO mrkj USER mr USING 1#
```

（2）创建具有题目中逻辑的存储过程：

```
CREATE PROCEDURE mr.proc_if_else(OUT text1 char(40))
BEGIN
    IF(SELECT AVG(成绩) FROM mr.学生表 x,mr.课程表 k,mr.成绩表 c
WHERE x.学号=c.学号 AND c.课程号=k.课程号 AND k.课程名='计算机应用技术')<73
THEN SET text1='平均成绩低于 73';
ELSE
    IF(SELECT AVG(成绩)FROM mr.学生表 x,mr.课程表 k,mr.成绩表 c
WHERE x.学号=c.学号 AND c.课程号=k.课程号 AND k.课程名='计算机应用技术')>=73
THEN SET text1='平均成绩大于等于 73';
END IF;
END IF;
END #
```

（3）调用上面的存储过程：

```
CALL mr.proc_if_else(?)#
```

上面嵌套的 IF…ELSE 语句等效于例 11.18 中非嵌套的 IF…ELSE 语句。在“命令编辑器”中执行上面命令后，返回结果与例 11.18 相同。

若子查询跟随在=、!=、<、<=、>、>=之后，或子查询用作表达式，子查询返回的值不允许多于一个。

2. CASE 语句

SQL PL 支持两种类型的 CASE 语句，以根据一个条件的状态实现逻辑的分支：

☑ 简单 CASE 语句用于根据一个字面值进入某个逻辑。

☑ 搜索型 CASE 语句用于根据一个表达式的值进入某个逻辑。

（1）简单 CASE 语句。

语法格式为：

```
CASE input_expression
    {WHEN when_expression THEN result_expression} [,…n]
    [ELSE else_result_expression]
END
```

参数说明如下。

☑ 计算 input_expression 表达式的值，并与每一个 when_expression 表达式的值进行比较，若相等，则返回对应的 result_expression 表达式的值；否则返回 else_result_expression 表达式的值。

☑ 参数 input_expression 和 when_expression：数据类型必须相同，或者可隐性转换。

☑ [,…n]：表示可以使用多个 WHEN when_expression THEN result_expression 子句。

（2）搜索型 CASE 语句。

语法格式为：

```
CASE
    {WHEN Boolean_expression THEN result_expression}[,...n]
    [ELSE else_result_expression]
END
```

参数说明如下。

☑ 按指定顺序为每个 WHEN 子句的 Boolean_expression 表达式求值，返回第一个取值为 TRUE 的 Boolean_expression 表达式对应的 result_expression 表达式的值；如果没有取值为 TRUE 的 Boolean_expression 表达式，则当指定 ELSE 子句时，返回 else_result_expression 之值；若没有指定 ELSE 子句，则返回 NULL。

☑ Boolean_expression：为布尔表达式，result_expression、else_result_expression 可为任意有效的 SQL Server 表达式。

【例 11.20】 在“学生表”中查询“学号”和“性别代号”。“性别代号”的值遵循下列规则：如果“学生表”中“性别”列的值为“男”，则“性别代号”的值为“1”；如果“学生表”中“性别”列的值为“女”，则“性别代码”的值为“0”（**实例位置：光盘\TM\sl\11\18**）。

实现步骤如下。

（1）连接到 MRKJ 数据库：

```
CONNECT TO mrkj USER mr USING 1#
```

（2）对学生按“性别”的值显示相应信息：

```
SELECT 学号,
(CASE 性别
WHEN '男' THEN 1
  WHEN '女' THEN 0
END) 性别代码
FROM mr.学生表 #
```

本例运行结果如图 11.10 所示。

说明

这是一个使用简单 CASE 语句的例子。对学生按“性别”的值显示相应的信息。

【例 11.21】 在“学生表”中查询“学号”和“年龄评价”。其中，“年龄评价”的值由下列逻辑确定：如果“年龄”小于 18 岁，则“年龄评价”的值为“年龄偏小”，如果年龄介于 18 到 25 岁之间，则“年龄评价”的值为“年龄适中”；如果年龄大于 25 岁，则“年龄评价”的值为“年龄偏大”（**实例位置：光盘\TM\sl\11\19**）。

实现步骤如下。

（1）连接 MRKJ 数据库：

```
CONNECT TO mrkj USER mr USING 1#
```

（2）根据学生的年龄范围显示相应信息：

```
SELECT 学号,
    (CASE
    WHEN YEAR(CURRENT DATE)-YEAR(出生日期)<18
    THEN '年龄较小'
    WHEN YEAR(CURRENT DATE)-YEAR(出生日期)BETWEEN 18 AND 25
    THEN '年龄适中'
ELSE '年龄偏大'
END) 年龄评价
FROM mr.学生表 #
```

本例运行结果如图 11.11 所示。

学号	性别代号
14001	0
14002	0
14003	1
14004	1
14005	0
14006	0
14007	0
14008	1
14009	0
14010	1
14969	1

图 11.10　显示相应信息

学号	年龄评价
14001	年龄偏大
14002	年龄偏大
14003	年龄偏大
14004	年龄偏大
14005	年龄适中
14006	年龄适中
14007	年龄偏大
14008	年龄适中
14009	年龄偏大
14010	年龄偏大
14969	年龄偏大

图 11.11　显示年龄范围的相应信息

说明

这是一个使用搜索型 CASE 语句的 SELECT 语句例子。查询语句能够根据学生的年龄范围显示相应信息。

11.6.2　迭代语句与退出循环语句

DB2 中的迭代语句主要包括 LOOP、WHILE、REPEAT 和 FOR 等 4 种，而退出循环语句主要包括 LEAVE 和 ITERATE 两种，下面分别进行讲解。

1. LOOP、WHILE、REPEAT 和 FOR 迭代语句

SQL PL 支持一些重复执行某个逻辑的方法，包括简单的 LOOP、WHILE 循环、REPEAT 循环和 FOR 循环，下面介绍这些迭代语句的语法。

（1）LOOP 循环——简单的循环。

```
L1:LOOP
SQL statements;
LEAVE L1;
END LOOP L1;
```

（2）WHILE 循环——进入前检查条件，当条件表达式的值为真时，执行构成循环体的 DB2 SQL 语句或语句块，然后再进行条件判断，重复上述操作，直至条件表达式的值为假，退出循环体的执行。

```
WHILE Boolean_expression
DO
    SQL statements
END WHILE;
```

（3）REPEAT 循环——退出前检查条件。

```
REPEAT
    SQL statements
    UNTIL Boolean_expression
END REPEAT;
```

（4）FOR 循环——结果集上的隐式循环。

```
FOR loop_name AS
    SELECT …FROM
DO
    SQL statements
END FOR;
```



注意

（1）FOR 语句不同于其他的迭代语句，因为它用于迭代一个定义好的结果集中的行。REPEAT 循环非常类似于 WHILE 循环，只不过条件是在最后进行检查的。

（2）上述各循环语句中的语句终止字符都是分号（；），不能改为“#”，因为上述各循环语句都是函数、存储过程或触发器的子句，上述各循环语句不能独立运行。子句中的语句终止字符可以是分号。

2. LEAVE 和 ITERATE 退出循环语句

SQL PL 中有两种跳出循环语句：LEAVE 语句和 ITERATE 语句。

LEAVE 语句的语法为：

```
LEAVE labelname;
```

一般用在循环语句中，用于退出本层循环。当程序中有多层循环嵌套时，使用 LEAVE 语句只能退出其所在的这一层循环。

ITERATE 语句的语法为：

```
ITERATE labelname;
```

一般用在循环语句中，结束本次循环，重新转到下一次循环条件的判断。

3．示例

为了演示这些循环技巧的使用，下面编写一组存储过程（例 11.22 至例 11.24）。

（1）要实现的存储过程的功能。这些存储过程功能都相同，存储过程从“学生表”中获取每个学生的“学号”、“姓名”和“年龄”，并将其插入到新表 STU 中。

注意

使用一个简单的 SQL 语句也可以做同样的事情，但是在这里使用 3 种不同的循环语句。

（2）创建示例表 STU。使用下面语句创建新表 STU：

```
CONNECT TO mrkj USER mr USING 1;
CREATE TABLE mr.stu
(
    学号 CHAR(8),
    姓名 CHAR(16),
    年龄 SMALLINT
)IN customdataspace1 ;
```

创建后的表如图 11.12 所示。

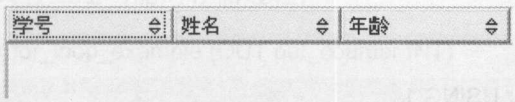


图 11.12　创建示例 STU 表

（3）实现上述功能的 3 个等价存储过程。

【例 11.22】　用 LOOP 循环创建具有上述功能的存储过程。用 LEAVE 退出其所在这一层的循环（**实例位置：光盘\TM\sl\11\20**）。

实现步骤如下。

（1）连接到 MRKJ 数据库：

```
CONNECT TO mrkj USER mr USING 1#
```

（2）使用 LOOP 循环创建具有上述功能的存储过程：

```
CREATE PROCEDURE mr.leave_loop(OUT out_counter INT)
    --out_cunter 输出插入到 STU 表中的记录总数
L1：BEGIN ATOMIC
    DECLARE var_at_end,var_counter INTEGER DEFAULT 0;
    --var_at_end   判断游标当前所指的位置是否是结果集的末尾
    --var_counter   记录已经插入 STU 表中的记录数
    DECLARE var_sid CHAR(8);          --指向学号的参数
    DECLARE var_name CHAR(16);        --指向姓名的参数
    DECLARE var_years_old INT;
    DECLARE c1 CURSOR FOR
```

```
SELECT 学号,姓名,
 (YEAR(CURRENT DATE)-YEAR(出生时间))年龄 FROM mr.学生表;
DECLARE CONTINUE HANDLER FOR NOT FOUND SET var_at_end =1;
    --如果发生找不到的异常情况则将 var_at_end 变成 1
OPEN c1;
FETCH_LOOP : LOOP
    FETCH c1 INTO var_sid,var_name,var_years_old;
IF var_at_end<>0 THEN --一直循环到游标所有指向的末一行
    LEAVE FETCH_LOOP;
    --如果不满足 IF 条件，则用 LEAVE 语句退出本层循环，即 FETCH_LOOP 标记的循环
END IF;
SET var_counter=var_counter+1;  --记录总数
INSERT INTO mr.stu
values(var_sid,var_name,var_years_old);
END LOOP FETCH_LOOP;
SET out_counter = var_counter;
END L1#
```

（3）调用上面含有 LOOP 循环的存储过程：

```
CALL mr.leave_loop(?)#
```

【例 11.23】 用 WHILE 循环创建具有上述功能的存储过程（**实例位置：光盘\TM\sl\11\21**）。实现步骤如下。

（1）连接 MRKJ 数据库：

```
CONNECT TO mrkj USER mr USING 1;
```

（2）为了执行与例 11.22 中存储过程相同的操作，需要先清空 mr.stu 表中的记录：

```
DELETE FROM mr.stu#
```

（3）用 WHILE 循环创建具有上述功能的存储过程：

```
CREATE PROCEDURE mr.while_example (OUT out_counter INT)
P1: BEGIN ATOMIC
    DECLARE var_at_end,var_counter INTEGER DEFAULT 0;
    --var_at_end 判断游标当前所指的位置是否是结果集的末尾
    --var_counter 记录已经插入 STU 表中的记录数
    DECLARE var_sid CHAR(6);          --指向学号的参数
    DECLARE var_name CHAR(16);        --指向姓名参数
    DECLARE var_years_old   INT;
    DECLARE c1 CURSOR FOR
    SELECT 学号,姓名,
        (YEAR(CURRENT DATE)-YEAR(出生日期)) 年龄 FROM mr.学生表;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET var_at_end=1;
    --如果发生找不到的异常情况则将 var_at_end 变成 1
    OPEN c1;
    FETCH c1 INTO var_sid,var_name,var_years_old;
    WHILE (var_at_end=0)
```

```
    DO
        INSERT INTO mr.stu
        values(var_sid,var_name,var_years_old);
        SET var_counter=var_counter+1;
    FETCH c1 INTO var_sid,var_name,var_years_old;
    END WHILE;
    SET out_counter=var_counter;
END p1#
```

（4）调用上面含有 WHILE 循环的存储过程：

```
CALL mr.while_example(?)#
```

【例 11.24】　用 FOR 循环创建具有上述功能的存储过程（**实例位置：光盘\TM\sl\11\22**）。

实现步骤如下。

（1）连接 MRKJ 数据库：

```
CONNECT TO mrkj USER mr USING 1;
```

（2）清空 mr.stu 表中的记录：

```
DELETE FROM mr.stu#;
```

（3）用 FOR 循环创建具有上述功能的存储过程：

```
CREATE PROCEDURE mr.for_loop_example (OUT out_counter INT)
P1:BEGIN ATOMIC
    DECLARE var_counter INT DEFAULT 0;
    FOR for_loop AS
        SELECT 学号,姓名,
    (YEAR(CURRENT DATE)-YEAR(出生日期))年龄 FROM mr.学生表
    DO
    INSERT INTO mr.stu
    values(for_loop.学号,for_loop.姓名,for_loop.年龄);
    SET var_counter = var_counter+1;
END FOR;
SET out_counter = var_counter;
END p1#
```

（4）调用上面的含有 FOR 循环的存储过程：

```
CALL mr.for_loop_example(?)#
```

说明

例 11.24 这个存储过程没有打开游标、从游标中读取数据或关闭游标，所有这些逻辑都是由 FOR 循环语句隐式进行的。而且可以引用循环中隐式地获取的值，使用选好名称限定列（例如 for_loop.学号），而不必使用本地变量来存储这些值。

11.6.3 RETURN 语句

RETURN 语句用于从过程、批处理或语句块中无条件退出，不执行位于 RETURN 之后的语句。RETURN 语句的语法格式为：

```
RETURN[integer_expression];
```



说明

integer_expression：返回整型表达式的值。存储过程可以为调用过程或应用程序返回整型值。



注意

除非特别指明，所有系统存储过程返回 0 值表示成功，返回非零值的表示失败。当用于存储过程时，RETURN 不能返回空值。

【例 11.25】 编写一个存储过程，以实现下列逻辑：检查某个学生的平均成绩，若大于 64，将返回状态代码 1，否则，将返回状态代码 2（**实例位置：光盘\TM\sl\11\23**）。

实现步骤如下。

（1）连接 MRKJ 数据库：

```
CONNECT TO mrkj USER mr USING 1;
```

（2）编写具有上述逻辑的存储过程：

```
CREATE PROCEDURE checkavg(IN param varchar(6))
BEGIN ATOMIC
    IF(SELECT AVG(成绩) FROM mr.成绩表 c
        WHERE c.学号= param GROUP BY 学号)>75
    THEN RETURN 1;
    ELSE RETURN 2;
    END IF;
END #
```

（3）调用上面定义的存储过程：

```
CALL checkavg('14006')
```

11.7 异常处理机制

11.7.1 声明有名称的条件

SQL PL 允许为给定的 SQLSTATE 声明用户命名的条件，以用于之后的错误处理。条件名称在整

个复合语句中必须是唯一的，并且只能在声明它的复合语句中引用它。

声明一个有名称的条件的语法为：

```
DECLARE condition_name CONDITION FOR
    SQLSTATE[VALUE] string_constant
```

【例 11.26】　举例说明条件声明的用法。

```
DECLARE foreign_key_violation CONDITION FOR SQLSTATE '14003'
DECLARE overflow CONDITION FOR SQLSTATE '14010'
```

11.7.2　声明条件处理程序

如果发生一个错误，存储过程的行为是根据条件处理程序来决定的。在一个存储过程中，可以为一个普通的或者有名称的条件和特定的 SQLSTATE 声明一个或多个条件处理程序。当一个 SQL 语句产生一个 SQLEXCEPTION 或 SQLWARNING （SQLCODE<>0）时，控制被转义到一个声明的处理程序中，以获取普通的异常或特定的 SQLSTATE 的值。

声明处理程序的语法为：

```
DECLARE <CONTINUE | EXIT | UNDO>HANDLER FOR
    <specific_condition_value | general_condition_value>SQL_procedure_statement
    WHERE specific_condition_value
    [SQLSTATE [VALUE] string_constant | condition_name]
```

在例 11.22 和例 11.23 中都包含了 CONTINUE HANDLER 语句，声明了异常条件处理程序。

11.7.3　强制发出异常

SQL PL 支持发出一个错误或警告条件。这导致一个具有指定 SQLSTATE 的错误或警告被返回，同时返回的还有可选的消息文本。

SIGNAL 语句的语法为：

```
SIGNAL [SQLSTATE [VALUE] <sqlstate_string_constant | variable_name | condition_name>]
[SET MESSAGE_TEXT =diagnostic_string_expression]
```

参数说明如下。

☑ diagnostic_string _expression：指定出现异常时的提示信息。

☑ 可以用字符串常量 sqlstate_string_constant 的形式发出一个用户定义的 SQLSTATE。其中，sqlstate_string_constant 包含 5 个字符，必须以数字 7，8，9 或者字母 I~Z 开始，其他字符必须来自数字集（0~9）或者字母 I~Z。

☑ 还可以发出一个特定的条件名称，但是必须在包含 SIGNAL 语句的复合语句中声明它，例如：

```
DECLARE condition overflow for SQLSTATE '14969';
......

SIGNAL overflow SET MESSAGE_TEXT='Too many characters,truncated';
```

11.8 MERGE 语句

在数据的转换和装载过程中，经常会用到 MERGE 语句。MERGE 语句也属于数据库操作语言（DML），使用 MERGE 语句可以将表中的数据合并到另一个表中，在合并的同时可以进行插入、删除、更新等操作。MERGE 的功能非常强大。MEGRE 语句的语法格式如下：

```
MERGE INTO <tableName | viewName | (fullselect)> [correlation_clause]
USING table_reference ON search_condition
WHEN matching_condition
THEN <modification_operation | signal_statement>
[ELSE IGNORE]
```

MEGRE 语法包括 5 个主要子句：

（1）MERGE 子句用于指定作为插入、更新或删除操作目标的表或视图。

（2）USING 子句用于指定要与目标连接的数据源。

（3）ON 子句用于指定决定目标与源的匹配位置的连接条件。

（4）WHEN 子句（WHEN MATCHED、WHEN NOT MATCHED BY TARGET 和 WHEN NOT MATCHED BY SOURCE）基于 ON 子句的结果和在 WHEN 子句中指定的任何其他搜索条件指定所要采取的操作。

（5）THEN 子句针对插入、更新或删除的目标执行插入、更新或删除操作。

参数说明如下。

- ☑ table_name | view_name | (fullselect)>：指定 MERGE 语句中更新、删除或插入操作的目标。如果 MERGE 操作的目标是全查询，则该全查询必须是可更新、删除或插入的，正如“可更新视图”“可删除视图”“可插入视图”一样。
- ☑ correlation_clause：可用于 search_condition 或 modification_operation 的 SET 语句中，用于指定一个表、视图或全查询的别名。
- ☑ USING table_reference：指定一组行作为结果表，如果表的表名为 table_reference，这个结果表要 MERGE 进目标。如果结果表为空，则返回警告。
- ☑ ON search_condition：指定 table_reference 中的哪些行将被用于 MERGE 语句中更新和删除操作，以及哪些行将被用于 MERGE 语句中插入操作。搜索条件将应用到目标表的每一行和 table_reference 表。对于 table_reference 表的那些行，如果 search_condition 的结果为真，则执行指定的更新或删除操作；对于 table_reference 表的那些行，如果 search_condition 的结果为假，则执行指定的插入操作。
- ☑ search_condition 有下列约束：

- 不能包含任何子查询、标量或其他。
- 不能包含任何没有被引用的操作或 DEREF 函数，也就是说，被引用的值不在目标标识列范围内。
- 不能包含 SQL 函数、XMLLQUERY 或 XMLEXISTS 表达式。
- 在搜索条件中被引用的任何列都必须是目标表、目标视图或 table_reference 中的列。
- 在全外连接的 JOIN 表达式中引用的任何函数必须是 DETERMINISTIC 和 NO EXTERNAL ACTION 的。

☑ WHEN matching_condition：指定执行 modification_operation 或 signal_statement 的条件。

☑ MATCHED：指定当 ON search_condition 为真时，在行集上要执行的操作。在 THEN 后面只能指定 UPDATE、DELETE 或 SIGNAL 语句。注意：这里的 UPDATE 或 DELETE 语句与完整的可单独执行的 UPDATE 或 DELETE 语句结构不同。

☑ NOT MATCHED：指定当 ON search_condition 为假或未知时，在行集上要执行的操作。在 THEN 后面只能指定 INSERT 或 SIGNAL 语句。注意：这里的 INSERT 语句与完整的可单独执行的 INSERT 语句结构不同。

☑ AND search_condition：指定与 ON search_condition 匹配的那些行上的进一步的搜索条件，这个进一步的搜索条件将用于 THEN 后面的操作。

☑ THEN modification_operation：指定当 matching_condition 的值为真时将执行的更新、插入或删除操作。

☑ ELSE IGNORE：指定如果没有使 matching_condition 为真的值，则不进行任何操作。

【例 11.27】　MERGE 语句的应用举例（**实例位置：光盘\TM\sl\11\24**）。

假设定义了一个雇员表和一个经理表，创建语句如下：

```
--雇员表--
create table employee(
empid integer not null,
name varchar(20) not null,
salary double
)#
--向雇员表中插入数据--
insert into employee (empid,name,salary) values
(1,'大毛',1500),
(2,'二毛','2200'),
(3,'三毛',1800),
(4,'四毛',1600),
(5,'阿毛',5000)#
--经理表—
create table manager(
maid integer not null,
name varchar(20) not null,
salary double
)#
--向经理表中插入数据—
insert into manager (maid,name,salary) values
```

```
(3,'三毛',5000),
(4,'四毛',6000),
(9,'周八',4000)#
```

你会发现，这样设计的表结构简直就是一个败笔，经理和雇员都是员工嘛，为什么要设计两个表呢？发现错误后就需要立即改正。所以你决定，删除经理表（manager），将 manager 表中的数据整合到 employee 表中，仔细检查发现，三毛和四毛在这两个表中都存在（可能人家干的好升官了），现在，我们要求把 employee 表中不存在的 manager 表中的记录，都插入到 employee 表中，已经存在的则更新薪水。该怎么办呢？这个问题并不难，我们可以分为两步，代码如下：

```
--更新存在的记录—
update employee   set salary =(select salary from manager where maid=empid )
where empid in(select maid from manager);

--插入不存在的记录--
insert into employee (empid,name,salary)
select maid,name,salary from manager where maid not in(select empid from employee);
```

通过以上的方法可以整合数据，但是我们还有更简单的方法，就是使用 MERGE 语句，代码如下：

```
merge into employee as em
using manager as ma
on em.empid=ma.maid
when matched then update set em.salary=ma.salary
when not matched then insert values (ma.maid,ma.name,ma.salary);
```

在上面的处理中，我们用经理表（manager）的薪水更新了雇员表（employee）的薪水，假设现在要求，如果经理表（Manager）的薪水>雇员表（employee）的薪水的时候更新，否则不更新，怎么办呢？具体代码如下：

```
--首先插入数据--
insert into manager (maid,name,salary) values
(1,'大毛',1500),-- 大毛干的好，也升官了，但是没有涨工资
(3,'三毛',5000),
(4,'四毛',6000),
(9,'周八',4000);

--如果经理表（Manager）的薪水>雇员表（employee）的薪水的时候更新，否则不更新—
merge into employee as em
using manager as ma
on em.empid =ma.maid
when matched and em.salary<ma.salary then update set em.salary =ma.salary
when not matched then insert values (ma.maid,ma.name,ma.salary);
```

大家有没有想过，对于雇员表中 salary 的数据>经理表中 salary 的数据的情况，理论上不可能存在。如果有，就说明有问题存在，这时如果想抛出异常，怎么办呢？具体代码如下：

```
merge into employee as em
using manager as ma
```

```
on em.empid = ma.maid
when matched and em.salary < ma.salary then update set em.salary=ma.salary
when matched and em.salary > ma.salary then signal sqlstate '70001' set message_text ='EM.SALARY>MA.
SALARY'
when not matched then insert values (ma.maid,ma.name,ma.salary)
ELSE IGNORE;
```

对于 EM.SALARY>MA.SALARY 的情况，如果你不想抛出异常，而是删除 EMPLOYEE 中的数据，怎么办呢？具体代码如下：

```
merge into employee as em
using manager as ma
on em.empid=ma.maid
when matched and em.salary < ma.salary then update set em.salary = ma.salary
when matched and em.salary > ma.salary then delete
when not matched then insert values (ma.maid,ma.name,ma.salary)
ELSE IGNORE;
```

11.9　小　　结

本章主要讲解了 SQL PL 语言、数据类型和变量、游标的概念和使用方法、流程控制语句、异常处理和 MERGE 语句的使用。读者使用游标提供的对一个结果集进行逐行处理的功能可以把结果集当成一个单元来处理。流程控制语句用于改变计算机的执行流程，以满足程序设计的需要。异常处理机制用于处理程序中的异常，从而保证程序的正常运行。

11.10　实践与练习

（1）DB2 中有哪些数据类型？（**答案位置：光盘\TM\sl\11\25**）

（2）举例说明声明变量的语句。（**答案位置：光盘\TM\sl\11\26**）

（3）列举出 SQL PL 语言元素。（**答案位置：光盘\TM\sl\11\27**）

第12章

DB2 中的函数

（视频讲解：38 分钟）

用户在编程时常常需要将一个或多个 SQL PL 语句组成子程序，以便反复调用。DB2 UDB 允许用户编写这些子程序：用户定义函数。本章将详细介绍 DB2 UDB 中系统内置函数的使用以及用户定义函数的创建、调用和删除。

通过阅读本章，您可以：

- 了解系统内置函数
- 掌握常用系统内置函数的使用
- 掌握创建和调用用户自定义函数的方法
- 熟悉用户定义函数的删除过程

12.1　系统内置函数

DB2 的系统内置函数有很多，要精通每一个函数几乎不可能，本节讲解 DB2 中常用的几种函数，以使读者对 DB2 系统内置函数有个初步的了解。

12.1.1　系统内置函数介绍

什么是函数？在 DB2 中，函数是一组封装的指令，用来执行一个特定的操作；可以使用一个或多个输入参数来定制操作，也可以使用一个或多个输出参数来返回结果。

DB2 中的 4 种函数类型如下：

☑ 标量函数。
☑ 聚集函数。
☑ 表函数。
☑ 行函数。

下面分别对这 4 种函数进行讲解。

1. 标量函数

标量函数是返回一个标量值的函数。可以使用标量函数执行简单的任务，也可以在标量函数中使用通过函数输入参数提供的值进行复杂的数学计算。标量函数的示例包括内置函数 LENGTH()函数和 SUBSTR()函数。

在 SQL 语句中支持表达式的任何地方都可以引用标量函数。当在查询谓词中使用时，标量函数可以改进总体性能，因为函数的逻辑作为引用它的 SQL 语句的组成部分在服务器上直接执行。另外，当在服务器上对一组候选行应用标量函数时，它可以起到过滤器的作用，限制必须返回给客户机的行数。但是，标量函数也有其局限性，例如，根据设计，标量函数只能返回一个值，无法返回多个值和结果集。另外，在标量函数中不支持事务管理，因此，在标量函数体中无法执行提交和回滚操作。

标量函数通常用来执行基本的数学计算和操作字符串。

2. 聚集函数

聚集函数（也称为列函数）也返回一个标量值，这个值是一组输入值的计算结果。通常情况下，这组输入值来自表中的一列，或来自 VALUES 子句中的元组。聚集函数的示例包括 MAX()函数和 MIN()函数。

3. 表函数

表函数向引用它的 SQL 语句返回一个表。只能在 SELECT 语句的 FROM 子句中引用表函数。但是，在能够应用于只读视图的任何操作（例如，连接、分组操作以及 UNION 和 INTERSECT 等集操作）

中，都可以使用表函数返回的表。

表函数可以发出操作系统调用、从文件中读取数据以及通过网络访问数据。另外，SQL 表函数还可以用来封装修改表数据的 SQL 语句（外部表函数不能封装 SQL 语句）。但是，与标量函数一样，在表函数中不支持事务管理；因此，在表函数体中无法执行提交和回滚操作。

表函数通常用来封装复杂但常用的子查询，以及为非关系数据提供表格式接口。例如，用户定义的外部表函数可以读取电子表格并生成一个数据表，然后可以把生成的数据表直接插入基表或通过查询访问它。

4．行函数

行函数返回单一表行。行函数只能供用户定义的结构化类型使用；不能单独使用行函数，也不能在抽象数据类型的上下文之外的 SQL 语句中使用。

通常，行函数用于把结构化类型属性映射到由内置数据类型值组成的行，以便能够在查询或 SQL 操作中访问结构化类型属性。例如，假设数据库有一个用户定义的结构化数据类型 Manager，此类型扩展另一个结构化数据类型 Employee，其中包含 Employee 和 Manager 特有的属性。如果希望在查询中引用属性值，那么可以创建一个行函数，它把属性值转换为查询可以引用的数据行。

12.1.2 常用系统内置函数

1．数学函数

数学函数可对 DB2 提供的数字类型数据（integer、smallint、bigint、real、decimal、double）进行数学运算，并返回运算结果。默认情况下，对 real 类型数据的内置运算的精度为 6 个小数位。默认情况下，传递到数学函数的数字将被解释为 decimal 数据类型，可用 CAST 或 CONVERT 函数将数学类型更改为其他数据类型。

下面是几个常用的数学函数及其用法说明。

（1）ABS(numeric_expression)

返回给定数字表达式的绝对值。参数 numeric_expression 为数字型表达式，返回值类型与 numeric_expression 相同。

【例 12.1】 在命令编辑器中运行以下命令（**实例位置：光盘\TM\sl\12\1**）。

```
VALUES ABS(-4.0),ABS(0.0),ABS(6.0);
```

本例运行结果如图 12.1 所示。

1
4.0
0.0
6.0

图 12.1 运行 ABS()

（2）ACOS(real_expression)

求一个余弦值对应的角度，角度值用弧度表示，本函数也称反余弦。参数 real_expression 为实型表达式，其取值范围从-1~1，对超过此范围的参数值，函数返回 NULL，并且报告域错误，返回值类型为 real。

（3）RAND([seed])

返回 0~1 之间的一个随机值。参数 seed 为整型表达式，返回值类型为 real。

2. 字符转换函数

下面是几个常用的字符转换函数及其用法说明。

（1）ASCII(string-expression)，返回字符表达式最左端字符的 ASCII 码值。在 ASCII()函数中，纯数字的字符串可不用单引号‘’括起来，但含其他字符的字符串必须用单引号‘’括起来使用，否则会出错。

（2）CHAR(integer_expression)，将 ASCII 码转换为字符。括号中的参数 integer_expression 为介于 0~255 之间的整数，如果没有输入 0~255 之间的 ASCII 码值，CHAR()返回 NULL。

（3）LOWER(string-expression)和 UPPER(string-expression)，LOWER()将字符串全部转换成小写；UPPER()将字符串全部转换成大写。

3. 去空格函数

下面是几个常用的去空格函数及用法说明。

（1）LTRIM(character_expression)，删除 character_expression 字符串中的前导空格，并返回字符串。参数 character_expression 为字符型表达式，返回值类型为 varchar。

（2）RTRIM(character_expression)，删除 character_expression 字符串尾部的空格，并返回字符串。参数 character_expression 为字符型表达式，返回值类型为 varchar。

4. 取字符串函数

下面是几个常用的取子串函数及其用法说明。

（1）LEFT(character_expression,integer_expression)，返回从字符串左边开始指定个数的字符。参数 character_expression 为字符型表达式，integer_expression 为整型表达式，返回值为 varchar 型。

【例 12.2】　返回课程名最左边的 8 个字符（**实例位置：光盘\TM\sl\12\2**）。

```
SELECT LEFT(课程名,8) FROM mr.课程表 ORDER BY 课程号
```

本例运行结果如图 12.2 所示。

（2）RIGHT(character_expression,integer_expression)，返回字符串 character_expression 右起 integer_expression 个字符。

【例 12.3】　返回课程名最右边的 44 个字符（**实例位置：光盘\TM\sl\12\3**）。

```
SELECT RIGHT(课程名,44) FROM mr.课程表 ORDER BY 课程号
```

本例运行结果如图 12.3 所示。

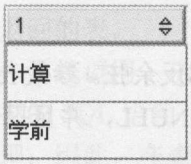


图 12.2　显示最左边的 8 个字符

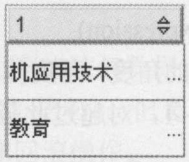


图 12.3　显示最右边 44 个字符

（3）SUBSTR(expression,start_position[,length])，返回 expression 中指定的部分数据。参数 expression 可为字符串、二进制串、text、image 字段或表达式；start_position、length 均为整型，前者指定子串的开始位置，后者指定子串的长度（要返回字节数）。如果 expression 是字符类型和二进制类型，则返回值类型与 expression 的类型相同。

（4）STR(real_expression[,length[,decimal_digits]])，把数值型数据转换为字符型数据。Length 指定返回的字符串的长度，decimal_digits 指定返回的小数位数。如果没有指定长度，默认的 length 值为 10，decimal_digits 默认值为 0。

当 length 或者 dicimal_digits 为负值时，返回 NULL；当 length 小于小数点左边（包括符号位）的位数时，返回 length 个“*”；先服从 length，再取 decimal_digits；当返回的字符串位数小于 length 时，左边补足空格。

5．字符串比较函数

下面是几个常用的字符串比较函数及其用法的说明。

（1）CHARINDEX(substring_expression,expression)，返回字符串中某个指定的子串出现的开始位置。其中 substring_expression 是所要查找的字符表达式，expression 可为字符串也可为列名表达式。如果没有发现子串，则返回 0 值。此函数不能用于 TEXT 和 IMAGE 数据类型。

（2）PATINDEX('%substring_expression%',column_name)，返回字符串中某个指定的子串出现的开始位置。其中子串表达式前后必须有百分号“%”，否则返回值为 0。与 CHARINDEX 函数不同的是，PATINDEX 函数的子串中可以使用通配符，且此函数可用于 CHAR、VARCHAR 和 TEXT 数据类型。

6．字符串操作函数

下面是几个常用的字符串操作函数及其用法说明。

（1）QUOTENAME(character_expression[,quote_character])，返回被特定字符括起来的字符串。其中，quote_character 标明括起来字符串所用的字符，默认值为“[]”。

（2）REPLICATE(character_expression,integer_expression)，返回一个重复 character_expression 指定次数的字符串。如果 integer_expression 值为负值，则返回 NULL。

（3）REVERSE(character_expression)，将指定的字符串的字符排列顺序颠倒。其中，character_expression 可以是字符串、常数或一个列的值。

（4）REPLACE(string_expression1,string_expression2,string_expression3)，返回被替换了指定子串的字符串。用 string_expression3 替换在 string_expression1 中的子串 string_expression2。

（5）SPACE(integer_expression)，返回一个由 integer_expression 指定长度的空白字符串。如果 integer_expression 值为负值，则返回 NULL。

（6）STUFF(character_expression1,start_position,length,character_expression2)，用另一子串替换字符串指定位置、长度的子串。如果起始位置为负或长度值为负，或者起始位置大于 character_expression1 的长度，则返回 NULL 值。如果 length 长度大于 character_expression1 中 start_position 右边的长度，则 character_expression1 只保留首字符。

7．数据类型转换函数

下面是几个常用的数据类型转换函数及其用法说明。

（1）CAST(expression AS data_type)：将 expression 转换为 data_type 数据类型，data_type 可以是 DB2 系统数据类型，也可以是用户定义的单值数据类型或用户定义的结构数据类型。但要注意 expression 与 data_type 的匹配，也就是说要保证 expression 能转换为 data_type 数据类型，否则会报错。

【例 12.4】　将“课程表”中的“成绩”作为 DEC(5,2)类型返回（**实例位置：光盘\TM\sl\12\4**）。

```
SELECT 学号,课程号,CAST(成绩 AS DEC(5,2))
FROM mr.成绩表
ORDER BY 课程号
```

本例运行结果如图 12.4 所示。

学号	课程号	3
14001	101	89.00
14002	101	83.00
14003	101	92.00
14004	101	84.00
14005	101	97.00
14007	102	64.00
14008	102	68.00
14009	102	62.00
14010	102	60.00

图 12.4　运行结果

“DEC(5.2)”表示总长 5 位（不包含小数点）且小数点后有 2 位数字的十进制小数。

（2）COALESCE(expression[,…n])，所有表达式必须是相同的类型，或者可以隐性转换为相同的类型，返回参数表中第一个非空表达式的值，如果所有自变量均为 NULL，则 COALESCE 返回 NULL 值。参数 expression 可为任何类型的表达式。n：表示可以指定多个表达式。

COALESCE (expression1,…n)与如下形式的 CASE 函数等价：

```
CASE
    WHEN(expression1 IS NOT NULL)THEN expression1
……
```

```
WHEN(expression1 IS NOT NULL)THEN expressionN
ELSE NULL
```

【例 12.5】 将“课程表”中的“成绩”的值转换为带 3 个小数位的格式并返回（**实例位置：光盘\TM\sl\12\5**）。

```
SELECT 学号,课程号,COALESCE (成绩,0.000) AS 成绩
    FROM mr.成绩表;
```

本例运行结果如图 12.5 所示。

学号	课程号	成绩
14001	101	89.000
14002	101	83.000
14003	101	92.000
14004	101	84.000
14005	101	97.000
14007	102	64.000
14008	102	68.000
14009	102	62.000
14010	102	60.000

图 12.5 显示 3 位小数

8. 日期函数

下面是几个常用的日期函数及其用法说明。

（1）DAY(date_expression)：返回 date_expression 中的日期值。

（2）MONTH(date_expression)：返回 date_expression 中的月份值。

（3）YEAR(date_expression)：返回 date_expression 中的年份值。

（4）DATEADD(dtepart_expression,number,date_expression)：返回指定日期 date_expression 加上指定的额外日期间隔 number 产生的新日期。

（5）DATEDIFF(datepart_expression,date_expression1,date_pression2)：返回两个指定日期在 datepart_expression 方面的不同之处，即 date_expression2 超过 date_expression1 的差距值，其结果值是一个带有正负号的整数值。

（6）DATENAME(datepart_expression,date_expression)：以字符串的形式返回日期的指定部分，此部分由 datepart_expression 来指定。

（7）DATEPART(datepart_expression,date_expression)：以整数值的形式返回日期的指定部分，此部分由 datepart_expression 来指定。

☑ DATEPART(dd,date_expression)等价于 DAY(date_expression)。

☑ DATEPART(mm,date_expression)等价于 MONTH(date)。

☑ DATEPART(yy,date_expression)等价于 YEAR(date_expression)。

（8）GETDATE()：以 DATETIME 的默认格式返回系统当前的日期和时间。

12.2　用户定义函数

用户定义函数（UDF）是用来扩展和增强 DB2 内置函数提供的功能的特殊对象。顾名思义，用户定义函数（UDF）是具有适当权利和/或特权的数据库用户创建的。与 DB2 内置函数不同，UDF 可以使用系统调用和 DB2 的管理 API，因此能够在系统、应用程序和数据库之间进行更多的协作。

可以创建 5 种 UDF：

☑ 有源（或模板）函数。

☑ SQL 标量、表或行函数。

☑ 外部标量函数。

☑ 外部表函数。

☑ OLE DB 外部表函数。

外部函数是使用 C、C++或 Java 等高级语言编写的返回单一值的函数。创建和实现有源函数或 SQL 函数的过程非常简单，但创建和使用外部函数的过程就复杂多了。本书只讲有源（或模板）函数及 SQL 标量、表或行函数。

通过执行 CREATE FUNCTION SQL 语句创建（或重新创建）UDF。这个语句有几种形式，应该使用的形式由要创建的函数的类型决定。

UDF 主要由以下部分组成：

☑ 函数名。

☑ 一系列参数声明（如果有参数的话）。

☑ 函数返回值的声明。

☑ 一个或多个函数选项。

☑ 函数体。

12.2.1　创建和调用用户定义函数

1. 有源（或模板）函数

有源函数是从一个已经向数据库注册的函数（称为源函数）构造出的 UDF。有源函数在本质上可以是标量、列或表函数，还可以用它们覆盖+、-、*、/等操作符。在调用有源函数时，传递给它的所有参数被转换为底层源函数所需的数据类型，然后执行原函数本身。执行完成时，源函数对产生结果执行必要的转换，并把结果返回给调用 UDF 的 SQL 语句。

有源函数最常见的用途是，让用户定义的不同数据类型有选择性地继承内置数据类型的某些语义。

通常，如果以现有函数作为源 UDF 的目的是向用户定义数据类型提供该函数的支持，那么 UDF 和有源函数使用相同的名称。这使用户能够用一个用户定义的不同类型调用相同的函数，而不需要额外的函数定义。一般情况下，多个函数可以使用相同的函数名，只要每个函数的签名都有些差异即可。

用来创建有源函数的 CREATE FUNCTION 语句的基本语法如下：

```
CREATE FUNCTION function_name ({parameter_name   input_datatype}[,...n])
    RETURNS output_datatype
    [SPECIFIC specific_name]
    <source_clause | as_template_clause>
```

其中，

```
source_clause ::=
    SOURCE [source_function_name [source_datatype] | SPECIFIC source_specific_name]
    [PARAMETER CCSID [ASCII | UNICODE ]]
as_template_clause ::=
    AS TEMPLATE
    [NOT DETERMINISTIC | DETERMINISTIC]
    [NO EXTERNAL ACTION | EXTERNAL ACTION]
```

参数说明如下。

- ☑ function_name：指定要创建的有源函数的名称。
- ☑ parameter_name：指定一个或多个函数参数的名称。
- ☑ intput_datatype：指定 parameter_name 所识别的参数所需的数据类型。
- ☑ output_datatype：指定函数返回的数据的类型。
- ☑ specific_name：指定分配给这个 UDF 的特定名称。这个名称可以用来引用或删除函数；但是，不能用来调用函数。在使用这种形式的 CREATE FUNCTION 语句时，如果指定了 specific_name，就会给创建的 UDF 分配一个唯一的名称。
- ☑ source_function_name：指定用来创建这个有源函数的现有函数。
- ☑ source_datatype：指定现有函数的每个参数期望接收的数据类型。
- ☑ source_specific_name：指定用来创建这个有源函数的现有函数的特定名称。
- ☑ 如果指定了 AS TEMPLATE 子句，那么会生成一个函数模板。函数模板是一个不完整的函数，它定义了要返回的数据类型，但是不包含可执行的代码。在联邦系统之中，函数模板被映射到一个数据源（如 Oracle 或 SQL Server）函数，这样就可以从联邦数据库调用数据源函数。只能在指定为联邦服务器的数据库服务器上注册函数模板。
- ☑ DETERMINISTIC | NOT DETERMINISTIC：表示在用相同的参数值调用函数时，函数是否总是返回相同的结果（DETERMINISTIC 代表确定性函数，NOT DETERMINISTIC 表示非确定性函数）。如果没有指定这两个子句，那么在用相同的参数值调用函数时，函数可能返回不同的结果。

【例 12.6】 基于内置数据类型 INTEGER 创建一个名为“AGE_TYPE”的数据类型，然后创建一个有源函数 AVG()，它接收并返回 AGE_TYPE 类型的值，而且它基于内置函数 AVG()（此函数接收并返回 INTEGER 值）（**实例位置：光盘\TM\sl\12\6**）。

实现步骤如下：

（1）连接到 MRKJ 数据库：

```
CONNECT TO mrkj USER mr USING 1;
```

（2）创建 mr.age_type 用户并定义单值类型，该单值类型是 INTEGER 的同义词：

```
CREATE DISTINCT TYPE mr.age_type AS INTEGER WITH COMPARISONS ;
```

（3）给 mr.age_type 用户定义单值类型，添加注释：

```
COMMENT ON DISTINCT TYPE mr.age_type IS '年龄类型';
```

（4）创建基于内置函数 SYSIBM.AVG(INTEGER)的有源函数 AVG(age_type)：

```
CREATE FUNCTION mr.AVG(mr.age_type) RETURNS mr.age_type
SOURCE SYSIBM.AVG(INTEGER);
```

2. 创建和调用 SQL 函数（包含 SQL 标量函数、表函数和行函数）

SQL 函数是指使用过程式 SQL 语句构造的 UDF。SQL 函数可以是标量函数，也可以返回一行或整个表。

（1）创建 SQL 函数的语法。用来创建 SQL 函数的 CREATE FUNCTION 语句的基本语法如下：

```
CREATE FUNCTION function_name({parameter_name input_datatype}[,…n])
RETURNS
<output_datatype | TABLE ({column_name   column_datatype}[,…n] | ROW ({column_name   column_datatype}
[,…n]))>
    [SPRCIFIC   specific_name]
        [LANGUAGE SQL]
        [PARAMETER CCSID[ASCII | UNICODE]]
        [DETERMINISTIC | NOT DETERMINISTIC]
        [EXTERNAL ACTION | NO EXTERNAL ACTION]
        [CONTAINS SQL | READS SQL DATA | MODIFIES SQL DATA]
        [STATIC DISPATCH]
        [CALLED ON NULL INPUT]
        [INHERIT SPECIAL REGISTERS]
        [PREDICATES (predicate_specification)]
        [INHERIT ISOLATION LEVEL<WITHOUT | WITH>LOCK REQUEST]
SQL_function_body
```

其中，

```
SQL_function_body ::=
    RETURN return_statement | dynamic_compound_statement
dynamic_compound_statement ::=
    [label_name :] BEGIN ATOMIC
    [local_variable_declaration | condition_declaration]
    <SQL_routine_statement | dynamic_compound_statement>
    END[label_name]
```

参数说明如下。

- ☑ SQL 复合语句是一种可以把多个 SQL 语句组合成一个可执行块的特殊机制。尽管复合语句中使用的每个 SQL 语句都可以单独执行，但是把它们组合在一起会减少处理它们所需的开销。在 SQL 函数体中使用 SQL 复合语句（要符合 SQL PL 规范）时，可以执行更复杂的逻辑。
- ☑ dynamic_compound_statement：SQL 动态复合语句，由一个或多个 SQL 语句组成。动态复合语句可以嵌套。在动态复合语句中的 SQL 函数或 SQL 方法中，可以使用 RETURN 语句。动态复合语句以关键字 BEGIN 和 END 界定。这些语句可以是 ATOMIC 或 NOT ATOMIC 的，默认情况下是 NOT ATOMIC。有两种类型的动态复合语句：NOT ATOMIC（默认）和 ATOMIC。
  - ➢ NOT ATOMIC：如果发生一个未处理的复合语句中的错误条件，不回滚任何 SQL 语句。
  - ➢ ATOMIC：如果发生任何未处理的复合语句中的错误条件，执行到这个点的所有可执行语句全部回滚。ATOMIC 语句中不能嵌套其他 ATOMIC 复合语句。
- ☑ SQL_routine_statement：SQL 常规复合语句，由一个或多个 SQL 语句组成，指定在 SQL 动态复合语句中可以使用的 SQL 语句。SQL 常规复合语句不支持在昵称上的搜索型更新、搜索型删除、搜索型插入或 MERGE 操作。
- ☑ SQL_routine_statement 中可以包含下列一个或多个元素：
  - ➢ CALL 语句。
  - ➢ DECLARE Variable 语句。
  - ➢ SET Variable 语句。
  - ➢ WHILE 循环。
  - ➢ FOR 循环。
  - ➢ IF 语句。
  - ➢ SIGNAL 语句。
  - ➢ ITERATE 语句。
  - ➢ LEAVE 语句。
  - ➢ GET DIAGNOSTIC 语句。
  - ➢ MERGE。
  - ➢ 全查询。
  - ➢ 搜索型删除（searched-delete）。
  - ➢ 搜索型更新（searched-update）。
  - ➢ 搜索型插入（searched-insert）。

如果 SQL_function_body 是动态复合语句，它必须包含至少一个 RETURN 语句，当函数被调用时必须执行 RETURN 语句。如果函数是一个表函数或行函数，它只能包含一个 RETURN 语句，且 RETURN 语句必须是动态复合语句的最后一条语句。

- ☑ Function_name：指定要创建的 SQL 函数的名称。
- ☑ Parameter_name：指定一个或多个函数参数的名称。
- ☑ Input_datatype：指定 parameter_name 所识别的参数所需的数据类型。
- ☑ Output_datatype：指定函数返回的数据的类型。

- ☑ 如果 RETURNS 后是 output_datatype，则表示这是个 SQL 标量函数；如果 RETURNS 后是 TABLE，则表示这是个 SQL 表函数；如果 RETURNS 后是 ROW，则表示这是个 SQL 行函数。
- ☑ column_name：指定函数返回的一列或多列的名称（如果此函数返回表或行的话）。
- ☑ column_datatype：指定 column_name 所识别的列返回的数据类型。
- ☑ specific_name：指定分配给这个 UDF 的特定名称。这个名称可以用来引用或删除函数；但不能用来调用函数。
- ☑ Return_statement：指定用于返回调用函数的应用程序的 RETURN SQL 语句（如果 SQL 函数体由动态复合语句组成，那么它必须包含至少一个 RETURN 语句；在调用函数时，必须执行一个 RETURN 语句。如果函数是表函数或行函数，那么只能包含一个 RETURN 语句，而且此语句必须是使用的最后一个语句）。
- ☑ LANGUAGE SQL：指定是用 SQL 编写的。
- ☑ EXTERNAL ACTION | NO EXTERNAL ACTION：表示函数执行的操作是否会改变不由 DB2 管理的对象的状态（EXTERNAL ACTION 表示会改变，NOEXTERNAL ACTION 表示不改变）。外部操作包括发送电子邮件或在外部文件中写记录等。如果没有指定这两个子句，那么意味着函数可能执行某种外部操作。
- ☑ CONTAINS SQL | READS SQL DATA | MODIFIES SQL DATA：表示 UDF 体中编写的 SQL 语句的类型。有 3 个值可用。
  - ➢ CONTAINS SQL：UDF 体包含的可执行 SQL 语句既不读数据，也不修改数据。
  - ➢ READS SQL DATA：UDF 体包含的可执行 SQL 语句读数据，但是不修改数据。
  - ➢ MODIFIES SQL DATA：UDF 体包含的可执行 SQL 语句既可以读数据，也可以修改数据。
- ☑ STATIC DISPATCH：表示在函数解析时 DB2 根据函数参数的静态类型（声明的类型）选择函数。
- ☑ CALLED ON NULL INPUT：表示无论任何参数是否包含 NULL 值，都调用此函数。
- ☑ INHERIT SPECIAL REGISTERS：指定函数中的可更新专用寄存器将从调用语句的环境中继承它们初始的值。对于被游标中的查询语句调用的函数，当游标被打开时，初始值继承于环境。对于嵌套对象中的例程（如触发器或视图），初始值继承于运行环境（而不是对象的定义）。专用寄存器的值不返回到函数的调用者。一些特殊寄存器，如日期时间寄存器，反映了当前正在执行的语句的属性，因此并不继承于调用者。
- ☑ PREDICATES：对于使用这个函数的谓词，这个子句标识具有下面属性的谓词，在谓词的搜索条件中可以利用索引扩展，且可以使用可选的 SELECTIVITY 子句。如果指定了 PREDICATES 子句，函数必须有 DETERMINISTIC 和 NO EXTERNAL ACTION 子句，如果指定了 PREDICATES 子句且数据库不是 UNICODE 数据库，则不能指定 PARAMETER CCSID UNICODE 子句。
- ☑ INHERIT ISOLATION LEVEL WITHOUT LOCK REQUEST | INHERIT ISOLATION LEVEL WITH LOCK REQUEST：指定当函数继承调用语句的隔离级别时，是否锁请求与调用语句的隔离子句关联。默认情况是 INHERIT ISOLATION LEVEL WITHOUT LOCK REQUEST。
  - ➢ INHERIT ISOLATION LEVEL WITHOUT LOCK REQUEST：指定当函数继承 SQL 调用语句的隔离级别时，如果 SQL 调用语句的上下文中的隔离级别子句中含有锁请求子句，

则函数不能被调用进调用语句的上下文中。

- INHERIT ISOLATION LEVEL WITH LOCK REQUEST：指定当函数继承 SQL 调用语句的隔离级别时，函数也继承了调用语句的锁请求子句。

（2）调用 SQL 函数的语法。调用 SQL 函数的方式在一定程度上取决于函数返回值的类型。如果函数是标量函数，就可以在 SELECT、VALUES 或 SET 语句中调用它。如果函数是表函数，那么必须在 SELECT 语句的 FROM 子句中引用它。在执行 SELECT 语句时，返回一个结果集。

调用表函数的语法格式为：

```
SELECT column_list
 FROM TABLE ([table_function_schema]table_function_name(parameter_list))
```

【例 12.7】 创建一个 SQL 标量函数，其函数体包含一条 SQL 语句，返回一个标量数据类型的值。此函数用于返回当前月数（**实例位置：光盘\TM\sl\12\7**）。

实现步骤如下。

（1）连接到 MRKJ 数据库：

```
CONNECT TO mrkj USER mr USING 1;
```

（2）创建用于定义 SQL 标量函数：

```
CREATE FUNCTION mr.this_month()
RETURNS VARCHAR(12)
LANGUAGE SQL
NOT DETERMINISTIC
NO EXTERNAL ACTION
CONTAINS SQL
RETURN MONTHNAME(CURRENT_DATE);
```

（3）调用上面创建的用户定义 SQL 标量函数：

```
VALUES mr.this_month();
```

返回结果如图 12.6 所示。

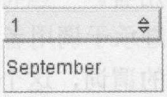


图 12.6 返回当前月份

上面的语句等效于下面的语句：

```
CONNECT TO mrkj USER mr USING 1;
VALUES MONTHNAME(CURRENT_DATE);
```

【例 12.8】 为结构数据类型“学生结构类型”定义一个转换函数。把“学生结构类型”的结构数据转换成一个数据行（**实例位置：光盘\TM\sl\12\8**）。

```
CONNECT TO mrkj USER mr USING 1;
CREATE FUNCTION mr. student_type (stu 学生结构类型)
```

```
RETURNS  ROW (学号 CHAR(6),姓名 CHAR(16),性别 CHAR(4))
LANGUAGE SQL
CONTAINS SQL
NO EXTERNAL ACTION
DETERMINISTIC
RETURN VALUES (stu..学号,stu..姓名,stu..性别);
```

12.2.2　用户定义函数的删除

对于一个已创建的用户定义函数，可有两种删除方法：

（1）通过“控制中心”删除。

（2）利用 SQL 语句删除。

```
DROP FUNCTION function_name;
```

或

```
DROP SPECIFIC function specific_function_name;
```

参数说明如下。

☑ function_name：指要删除的用户定义的函数名称，包含模式名。可以选择是否指定所有者的名称，但不能指定服务器名称和数据库名称。

☑ specific_function_name：指要删除的用户定义的函数的特定名称。

12.3　小　　结

本章主要讲解 DB2 数据库系统的内置函数，以及用户自定义函数的使用，学习本章内容时，读者应该重点掌握常用的系统内置函数的使用，并熟练掌握如何创建和调用用户自定义的函数。

12.4　实践与练习

（1）说明系统内置函数的分类及各类函数的特点。（**答案位置：光盘\TM\sl\12\9**）

（2）什么是用户自定义函数？（**答案位置：光盘\TM\sl\12\10**）

（3）列举出 5 种可以创建的用户自定义函数。（**答案位置：光盘\TM\sl\12\11**）

第13章

存储过程和触发器

（视频讲解：71分钟）

用户在编程时常常需要将一个或多个SQL PL语句组成子程序，以便反复调用。DB2 UDB允许用户编写这些子程序：存储过程和触发器。本章将详细介绍存储过程和触发器的定义和使用。

通过阅读本章，您可以：

- 熟悉存储过程的基本概念
- 熟悉存储过程的几种主要分类
- 熟练掌握存储过程的创建及执行
- 熟悉存储过程的重载、查询、修改和删除等操作
- 熟悉触发器的基本概念及分类
- 掌握如何创建、修改和删除触发器

13.1　存 储 过 程

在存储过程中，开发人员能够声明和设置 SQL 变量、实现流程控制、处理异常，并对数据进行插入、更新或者删除。同时，客户应用和存储过程之间可以传递参数，并且从存储过程中返回结果集。存储过程可以使用 C、C++、Java 等多种方法实现，其中，使用 SQL 编写的 DB2 存储过程是在开发中常见的一种存储过程。

13.1.1　存储过程的概念

存储过程是具有以下特征的程序：

（1）用 CREATE PROCEDURE 语句注册。

（2）包含一个或多个 SQL 语句和一些用编程语言编写的操作或逻辑处理。

（3）存储在数据库中且在 DB2 服务器上运行。

（4）可以由正在使用 SQL 的应用程序使用 CALL 语句来调用。

（5）允许应用程序分两部分运行：在客户机上运行应用程序，在服务器上运行存储过程。

图 13.1 说明了没有存储过程时，客户机应用程序如何存取位于数据库服务器上的数据库。所有数据库存取都必须通过网络来执行，在某些情况下，这可能会导致性能较差。对于每个 SQL 语句，数据库管理器应用程序必须启动与 DB2 的独立通信。

图 13.2 说明了客户机应用程序通过存储过程存取位于数据库服务器上的数据库。

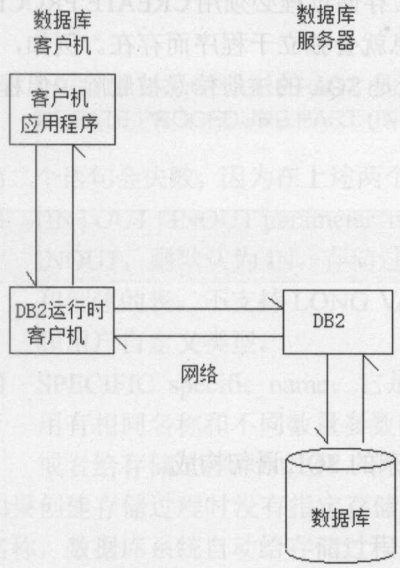


图 13.1　不使用存储过程的客户机应用程序

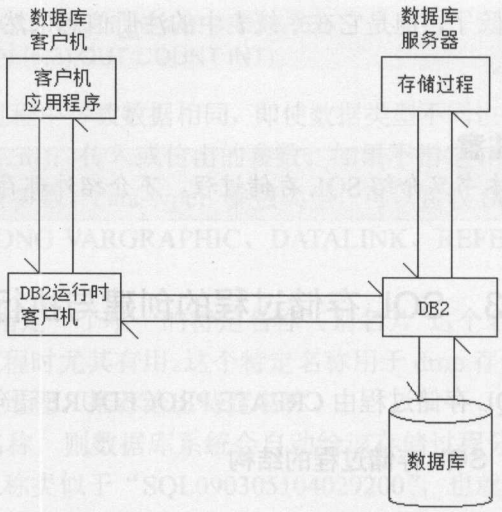


图 13.2　使用存储过程的客户机应用程序

存储过程的主要作用如下：

（1）要提高应用程序性能，可以创建在数据库服务器上运行的存储过程。存储过程能够帮助提高应用程序性能并减少数据库存取流量。客户机应用程序使用 CALL 语句来启动存储过程。当存储过程运行时，它在服务器上运行 SQL 语句，并存取服务器上的数据库，然后将所请求的信息返回给客户机应用程序。于是，客户机应用程序只需调用存储过程，就可以获取该存储过程中包含的 SQL 语句的结果。

（2）存储过程还可以帮助集中业务逻辑。如果对存储过程进行更改，则使用该存储过程的所有客户机应用程序都立即可以应用这些更改。

要使用存储过程，需要编写两个程序：存储过程（它在数据库服务器上运行）和客户机应用程序（它在客户机工作站或中间件服务器上运行，例如在 Web 服务器上运行）。客户机应用程序使用可用的其中一种 API 方法来调用存储过程。存储过程遵循与客户机应用程序交换数据的某些约定。

13.1.2 存储过程的类型

存储过程分为外部存储过程和 SQL 存储过程，下面分别进行介绍。

1. 外部存储过程

可以是任何高级语言的程序对象（除了 SYSTEM/36 的程序）和 REXX 过程。外部存储过程不必须包含 SQL 语句，但是允许包含 SQL 语句。虽然外部存储过程不必注册，但是大多数的外部存储过程都是注册过的。

用于编写 DB2（外部）存储过程的常用语言有：Assembler、C、C++、COBOL、Java、PL/I 和 REXX。

2. SQL 存储过程

SQL 存储过程是由 SQL 语句和控制 SQL 语句组成的。SQL 存储过程必须用 CREATE PROCEDURE 语句进行注册。一旦存储过程注册后，在系统索引表中的信息就会独立于程序而存在。例如，虽然程序被删除了，但是它在系统表中的注册信息依然存在。也可能是 SQL 的注册信息被删除了但程序却依然存在。



注意

本书只介绍 SQL 存储过程，不介绍外部存储过程。

13.1.3 SQL 存储过程的创建与执行

SQL 存储过程由 CREATE PROCEDURE 语句、参数和复杂的 SQL 语句构成。

1. SQL 存储过程的结构

```
CREATE PROCEDURE proc_name
[List of the IN/OUT/INOUT parameters]
```

```
[Procedure properties optional clauses]
[Generation optional clauses]
[SQL procedure body-compound statement]
```

2. CREATE PROCEDURE（SQL）命令语法

```
CREATE PROCEDURE procedure_name
({[IN | OUT | INOUT] parameter_name datatype}[,...n])
[SPECIFIC specific_name]
[DYNAMIC RESULT SETS [0 | integer]]
[MODIFIES SQL DATA | CONTAINS SQL | READS SQL DATA]
[NOT DETERMINISTIC | DETERMINISTIC] [CALLED ON NULL INPUT]
[INHERIT SPECIAL REGISTERS]
[OLD SAVEPOINT LEVEL | NEW SAVEPOINT LEVEL]
[LANGUAGE SQL] [EXTERNAL ACTION | NO EXTERNAL ACTION]
[PARAMETER CCSID ASCII | UNICODE]
SQL_procedure_body
```

其中：

```
SQL_procedure_body ::=
[label_name: ]BEGIN [ ATOMIC | NOT ATOMIC]
[local_variable_declaration] [cursor_declaration][condition_declaration][handler_declaration]
<SQL_routine_statement | dynamic_compound_statement>
END [label_name]
```

参数说明如下。

☑ procedure_name：存储过程的名字，最长允许 128 个字符，在同一个数据库的同一个模式下，不能存在存储过程名和参数数目都相同的存储过程，即使参数的类型不同也不行。例如下列语句。

```
CREATE PROCEDURE PART (IN NUMBER INT,OUT PART_NAME CHAR(35))
CREATE PROCEDURE PART (IN COST DECIMAL(5,3),OUT COUNT INT)
```

第二个语句会失败，因为在上述两个 PART 存储过程中参数数据相同，即使数据类型不同也是如此。

☑ (IN | OUT | INOUT parameter_name datatype[,..n])：传入或传出的参数。如果不指定 IN | OUT | INOUT，则默认为 IN。存储过程中可以没有参数。data-type：参数类型，可以接收 SQL 类型和创建的表。不支持 LONG VARCHAR、LONG VARGRAPHIC、DATALINK、REFERENCE 和用户自定义类型。

☑ SPECIFIC specific_name：它是给存储过程指定一个唯一的特定名称（别名），这个名称在使用有相同名称和不同数量参数的重载存储过程时尤其有用。这个特定名称用于 drop 存储过程，或者给存储过程添加注释，但不能调用存储过程。推荐给出特定名称。

如果创建存储过程时没有指定存储过程的特定名称，则数据库系统会自动给该存储过程分配一个特定名称，数据库系统自动给存储过程分配的特定名称类似于“SQL090305104029200”，也就是 SQL 后面连接上时间戳“yymmddhhmmsshhn”。

☑ DYNAMIC RESULT SETS integer：指定存储过程返回结果集的最大数量。存储过程中虽然没

有 RETURN 语句，但是却能返回结果集。

☑ CONTAINS SQL | READS SQL DATA | MODIFIES SQL DATA：指定存储过程中的 SQL 访问级别。

- CONTAINS SQL：表示存储过程既不可读取 SQL 数据，也不可修改 SQL 数据。
- READS SQL DATA：表示存储过程可读取 SQL，但不可修改 SQL 数据。
- MODIFIES SQL DATA：表示存储过程可以执行任何 SQL 语句。可以对数据库中的数据进行增加、删除和修改。

☑ DETERMINISTIC | NOT DETERMINISTIC：表示存储过程是动态的或者非动态的。动态存储过程返回值是不确定的（NOT DETERMINISTIC）；非动态存储过程的返回值是确定的（DETERMINISTIC），每次执行的返回值相同。

☑ CALLED ON NULL INPUT：表示可以调用存储过程，而不管输入参数是否为 NULL，并且任何的 OUT 或者 INOUT 参数可以返回一个 NULL 或者非空值。检验参数是否为 NULL 是在存储过程中进行的。

☑ INHERIT SPECIAL REGISTERS：表示继承专用寄存器。

☑ OLD SAVEPOINT LEVEL | NEW SAVEPOINT LEVEL：建立存储点。OLD SAVEPOINT LEVEL 是默认的存储点。

☑ EXTERNAL ACTION | NO EXTERNAL ACTION：EXTERNAL ACTION 表明存储过程能采取一些措施改变非数据库管理器托管的对象的状态（EXTERNAL ACTION）；NO EXTERNAL ACTION 则与之相反。默认情况是 EXTERNAL ACTION。

☑ SQL_procedure_body：SQL 存储过程复合语句，它是存储过程的主体，它的核心是一组复合语句，复合语句以关键字 BEGIN 和 END 定界。这些语句可以是 ATOMIC 或 NOT ATOMIC 的，默认情况是 NOT ATOMIC。

3. 使用 CALL 语句调用存储过程

在 DB2 的命令编辑器中使用 CALL 语句调用存储过程。

使用 CALL 语句前需要连接至相应的数据库。要使用 CALL 语句，需要输入存储过程名，加上任何 IN 或 INOUT 参数，并输入“?”作为每个 OUT 参数的占位符。存储过程的参数是在任何源文件中创建存储过程的 CREATE PROCEDURE 语句中给定的。本书将在后面的示例中详细讲解。

存储过程传递参数有下面的规则：

（1）变量和参数必须严格匹配。

（2）所有的参数都必须有值（包括占位符“?”）。

（3）重载存储过程的区分和参数的匹配取决于参数的数目。

4. 示例

为了用尽可能少的示例讲解存储过程的使用方法，此处列举的每个存储过程示例都包括多个 SQL PL 元素的应用，例如参数、复合 SQL 语句、变量声明、游标、赋值语句、诊断语句、流程控制语句、异常处理语句等。

如在例 13.1 所示中，设计了一个 SQL 存储过程，用于删除已有的指定名称的用户存储过程。

【例 13.1】　设计一个 SQL 存储过程，查询系统中是否已经存在用户定义的名称为 pro_in 的存储过程，如果不存在该存储过程，则不做任何操作，如果存在，则将其删除。

说明

存储过程的所有定义都记录在 SYSCAT.PROCEDURES 系统目录视图中，如果该系统目录视图中存在名称为 pro_in 的存储过程的信息，则表示已经存在名称为 pro_in 的存储过程，需用 DROP PROCEDURE 语句删除该存储过程。

注意

在后面列举的很多存储过程中都调用了这里定义的“clear_proc”存储过程。

实现步骤如下。

（1）连接到 MRKJ 数据库：

```
CONNECT TO mrkj USER mr USING 1#
```

（2）创建存储过程 mr.clear_proc：

```
CREATE PROCEDURE mr.clear_proc
(IN proc_schema VARCHAR(128),IN pro_in VARCHAR(128))
SPECIFIC mr.clear_proc
BEGIN
DECLARE proc_count INTEGER default 0;
DECLARE statement VARCHAR(200);
SELECT COUNT(P.PROCNAME) INTO proc_count FROM syscat.procedures p
WHERE P.PROCNAME = UPPER (pro_in)
AND P.PROCSCHEMA =UPPER (proc_schema);
IF (proc_count = 1)
THEN SET statement =
'DROP PROCEDURE' || UPPER (proc_schema) || '.' || UPPER (pro_in);
EXECUTE IMMEDIATE statement;
RETURN 1;
END IF;
END#
```

（3）查看已创建的存储过程。

在“控制中心”对象视图中选择 MRKJ→“应用程序对象”→“存储过程”文件夹，在“控制中心”右面的内容窗格中可以看到刚刚新建的 SQL 存储过程 CLEAR_PROC（存储过程名称已自动转换为大写形式）。在 CLEAR_PROC 单击鼠标右键，在弹出的快捷菜单中选择“显示 SQL 过程主体”命令，可以查看该存储过程的代码；如果选择“删除”命令，则可以删除该存储过程，如图 13.3 所示。

在命令编辑器中运行以下命令也可以查询到刚才创建的 CLEAR_PROC 存储过程，如图 13.4 所示。

```
select * from syscat.procedures where procname = upper ('clear_proc');
```

这里必须使用 upper 函数，否则系统会返回 0 个结果。因为系统已经将存储过程名称“clear_proc”

自动变成了大写“CLEAR_PROC”，存储在 syscat.procedures 系统视图中的存储过程名是大写后的“CLEAR_PROC”。

MR-PC - DB2 - MRKJ - 存储过程

名称	模式	特定名称
CLEANUP	SYSIB...	DBMS_ALERT_CLEANUP
CLEAR_PROC	MR	CLEAR_PROC
CLOSE_BLOB	SYSIB...	DBMS_LOB_CLOSE_BLOB
CLOSE_CLOB	SYSIB...	DBMS_LOB_CLOSE_CLOB

图 13.3　新创建并注册的用户存储过程 CLEAR_PROC

PROCSCHEMA	PROCNAME	SPECIFICNAME	PROCEDURE_ID	DEFINER	PARM_COUNT	PARM
MR	CLEAR_PROC	CLEAR_PROC	66522	MR	2	

图 13.4　查询已创建的 CLEAR_PROC 存储过程

（4）该存储过程的用途。

在创建其他 SQL 存储过程前调用上述 clear_proc 存储过程，可删除已存在的与将创建的存储过程同名的存储过程。例如：

```
CONNECT TO mrkj USER mr USING 1;
CALL mr.clear_proc('mr','student_info');
```

或：

```
CONNECT TO mrkj USER mr USING 1;
CALL mr.clear_proc('MR','STUDENT_INFO');
```

如果已存在名为 mr.student_info 的存储过程，则将其删除。然后用户可以用 CREATE PROCEDURE 语句创建新的名称为 mr.student_info 的存储过程，系统会自动将小写的存储过程名称转换为大写，所以虽然 CREATE PROCEDURE 语句中所用的存储过程名大小写状态不同，但如果先创建名为 mr.student_info 的存储过程，再创建名为“MR.STUDENT_INFO”的存储过程，系统就会报错，因为它们会被认为是同名的。

在如例 13.2 所示中，建立包含简单 SQL 语句的 SQL 存储过程的示例，且不含任何参数，包含游标。

【例 13.2】　创建一个存储过程，它从 MRKJ 数据库的 3 个表中查询，返回学生的“学号”“姓名”“课程名”“成绩”“学分”（**实例位置：光盘\TM\sl\13\1**）。

实现步骤如下。

（1）连接到 MRKJ 数据库：

```
CONNECT TO mrkj USER mr USING 1#
```

（2）如果已存在名称为 mr.simple_sql_proc 的存储过程，则将其删除：

```
CALL mr.simple_sql_proc('mr','simple_sql_proc')#
```

（3）创建存储过程 mr.simple_sql_proc：

```
CREATE PROCEDURE mr.simple_sql_proc
BEGIN
DECLARE c2 CURSOR WITH RETURN FOR
SELECT s.学号,姓名,课程名,成绩,学分
FROM mr.学生表 s INNER  JOIN mr.成绩表 sc ON s.学号 =sc.学号
INNER JOIN mr.课程表 c ON sc.课程号 = c.课程号;
OPEN c2;
END#
```

（4）调用上面创建的存储过程：

```
call mr.simple_sql_proc()#
```

运行结果如图 13.5 所示。

结果集 1

学号	姓名	课程名	成绩	学分
14001	马芳	计算机应用技术	89	5
14002	宋静	计算机应用技术	83	5
14003	周天星	计算机应用技术	92	5
14004	蒋阳泉	计算机应用技术	84	5
14005	董敏	计算机应用技术	97	5
14007	秦冬雪	学前教育	64	8
14008	欧阳云	学前教育	68	8
14009	董淑芬	学前教育	62	8
14010	禁传奇	学前教育	60	8

9 条记录已选择。

返回状态 = 0

图 13.5　运行结果

创建包含简单 SQL 语句且带参数的存储过程，如例 13.3 所示。

【例 13.3】　将 MRKJ 数据库的学生表中“学号”为 in_sid 的学生“姓名”修改为 in_sname。该存储过程接收与传递参数精确匹配的值（**实例位置：光盘\TM\sl\13\2**）。

实现步骤如下。

（1）连接到 MRKJ 数据库：

```
CONNECT TO mrkj USER mr USING 1#
```

（2）如果已存在名称为 mr.proc_with_parameter 的存储过程，则将其删除：

```
CALL mr.proc_clear_proc ('mr','mr.proc_with_parameter')#
```

（3）创建存储过程 mr.proc_with_parameter：

```
CREATE PROCEDURE mr.proc_with_parameter(IN in_sid CHAR(6), IN in_sname CHAR(16))
SPECIFIC mr.proc_with_parameter
```

```
LANGUAGE SQL
UPDATE mr.学生表
SET 姓名=in_sname
WHERE 学号=in_sid#
```

（4）调用上面创建的存储过程：

```
call mr.proc_with_parameter ('14001','mafang')#
```

（5）根据本书需要，再次调用该存储过程，将学号为“14001”的学生姓名改回“马芳”。

```
call mr.proc_with_parameter ('14001','马芳')#
```

创建使用复合 SQL 语句的 SQL 存储过程示例，返回结果中包含 RESULT SETS，如例 13.4 所示。

【例 13.4】 定义一个 SQL 存储过程，给它输入课程名 in_cname，则返回该门课程的平均成绩，并且用一个结果集返回选修了该门课程且超过平均成绩的所有学生的分数，结果集包括“学号”“姓名”“课程号”“课程名”“成绩”（**实例位置：光盘\TM\sl\13\3**）。

实现步骤如下。

（1）连接到 MRKJ 数据库：

```
CONNECT TO mrkj USER mr USING 1#
```

（2）如果已存在名称为 mr.median_result_set 的存储过程，则将其删除：

```
CALL mr.proc_clear_proc ('mr','mr.median_result_set')#
```

（3）创建存储过程 mr.median_result_set：

```
CREATE PROCEDURE mr.median_result_set (IN in_cname CHAR(40), OUT average_score DOUBLE)
SPECIFIC mr. median_result_set   --特定名称
RESULT SETS 1    --返回的结果集的数目
LANGUAGE SQL   --指定程序的主体用的是 SQL 语言
P1:BEGIN    --SQL 存储过程主体部分的开始标识符 BEGIN;
            --BEGIN 前面还可以加上行标（Label）和冒号，推荐使用行标;
            --BEGIN 后面还可以加上 ATOMIC/NOTATOMIC，默认是 ATOMIC
            --本地变量声明(local-variable-declaration)
            DECLARE v_numRecords INTEGER DEFAULT 1;
            DECLARE v_counter INTEGER DEFAULT 0;
            --声明游标(cusor-declaration)
            --c1 返回课程名为 in_cname 的课程的平均成绩
            DECLARE c1 CURSOR FOR
            SELECT CAT (成绩 AS DOUBLE)FROM mr.成绩表 sc,mr.课程表 c
            Where c.课程名=in_cname AND sc.课程号=c.课程号
            ORDER BY sc.课程号,成绩;

            --c2 是一个结果集，返回选修了课程名为 cname 的课程且超过平均成绩的所有学生的分数，
            --结果集包括学号、姓名、课程号、课程名和成绩字段
            DECLARE c2 CURSOR WITH RETURN FOR
            SELECT DISTINCT s.学号,s.姓名,sc.课程号,c.课程名,成绩
            FROM mr.学生表 s INNER JOIN mr.成绩表 sc ON s.学号=sc.学号
```

```
INNER JOIN mr.课程表 c ON sc.课程号=c.课程号
Where 成绩>average_score AND c.课程名=in_cname
ORDER BY sc.课程号,成绩;
--声明异常条件处理程序(handler-declaration)
DECLARE EXIT HANDLER FOR NOT FOUND
SET average_score=6666;
--此异常条件处理程序声明中包含一个赋值复合语句(Assignment compound statements)
--如果没有符合查询条件的记录，就返回 average_score=6666

SET average_score=0;  --赋值复合语句（Assignment compound statements）

--SQL 复杂语句，用 SQL 语句向整型参数 v_numRecords 传值
select distinct count(*) into v_numRecords
--查询语句中的 into 子句只能用于函数、存储过程、触发器
FROM mr.成绩表 sc,mr.课程表 c
Where 成绩 > average_score AND  c.课程名=in_cname AND sc.课程号=c.课程号;

--流控制复合语句(flow of control compound statements)
OPEN c1;
WHILE v_counter <(v_numRecords/2+1)
DO
FETCH c1 INTO average_score;
SET v_counter =v_counter+1;
END WHILE;
CLOSE c1;
OPEN c2;
--SQL 存储过程主体部分的结束标志符 END，END 后面还要加上程序结束符号分号（;）
END P1#
```

（4）调用上面创建的存储过程：

```
call mr.median_result_set ('计算机应用技术',?)#
```

执行结果如图 13.6 所示。

```
输出参数的值
------------------------
参数名：AVERAGE_SCORE
参数值：+8.90000000000000E+001


结果集 1
----------

学号    姓名      课程号 课程名                                成绩
------- --------- ------ ------------------------------------ -----
14003   某某      101    计算机应用技术                          92
14005   董敏      101    计算机应用技术                          97

2 条记录已选择。

返回状态 = 0
```

图 13.6　返回结果

创建包含会话全局变量和存储过程递归调用的 SQL 存储过程，如例 13.5 所示。

（1）使用 CREATE VARIABLE 语句创建会话全局变量共享数据。

DB2 支持会话全局变量，一个会话全局变量与某个特定会话相关，是被会话中每个存储过程应用的全局变量，包含一个值，该值在会话中唯一。

会话全局变量的语法为：

```
CREATE VARIABLE var_name DATATYPE [DEAFULT value]
```

这里需要注意的是，会话全局变量是在存储过程外面声明的。

（2）用 GET DIAGNOSTIC 语句获取 CALL 语句的返回值。

在嵌套存储过程中，RETURN 语句返回存储过程的调用控制流，从而立即终止一个 SQL 程序。RETURN 语句也可以将一个整数值传递给主调存储过程。如果没有提供任何值，那么默认值为 0。DB2 支持用 GET DIAGNOSTIC 获取刚刚被执行的 CALL 语句的信息。

GET DIAGNOSTIC 的语法为：

```
GET DIAGNOSTIC <returned_code>=DB2_RETURN_STATUS
```

注意

诊断语句所在的存储过程中需要声明一个参数，该参数用于接收 DB2_RETURN_STATUS 的值。

【例 13.5】 定义一个 SQL 存储过程 student_totalcredit，如果输入学生的姓名，则返回该学生的总分数。定义另一个存储过程 student_status，用于判定这个学生是否修够学分，且足以获得学士学位（假设修够 50 个学分即获得学士学位）（**实例位置：光盘\TM\sl\13\4**）。

实现步骤如下。

（1）创建存储过程首先需要连接数据库：

```
CONNECT TO mrkj USER mr USING 1#
```

（2）删除全局变量 global_var_totalcredit：

```
DROP VARIABLE mr.global_var_totalcredit#
--如果先前已经定义了变量 global_var_totalcredit 则删除该变量;
--如果先前没有定义变量 global_var_totalcredit 则返回错误
--但不影响后面其他 SQL 语句的执行
```

（3）创建会话全局变量 global_var_totalcredit：

```
CREATE VARIABLE mr. global_var_totalcredit INT default 0 #
```

（4）如果已经存在名称为 mr.student_totalcredit 的存储过程，则将其删除：

```
CALL mr.clear_proc('mr','student_totalcredit')#
```

（5）创建存储过程 mr. student_totalcredit：

```
CREATE PROCEDURE mr.student_totalcredit(IN var_sname CHAR(16))
--存储过程中包含一个参数 var_return，用于接收 RETURN 返回的状态值
```

```
BEGIN
    SELECT 总分数 INTO mr.global_var_totalcredit
      FROM mr.学生表 WHERE 姓名=var_sname;
      RETURN mr.global_var_totalcredit;
      --将被变量 var_diagnostics 诊断的返回状态值
END #
```

（6）如果已经存在名为 mr. student_status 的存储过程，则将其删除：

```
CALL mr.clear_proc('mr','student_status')#
```

（7）创建存储过程 mr.student_status，在它的复合 SQL 语句中调用了上面定义的存储过程 student_totalcredit：

```
CREATE PROCEDURE mr.student_status
(IN p_sname CHAR(6),OUT p_new_status CHAR(100),OUT var_diagnostics INTEGER)
--存储过程中包含一个参数 var_diagnostics，用于接收 DB2_RETURN_STATUS 的值
BEGIN ATOMIC
CALL mr.student_totalcredit(p_sname);--调用前面定义的存储过程 mr.student_totalcredit
GET DIAGNOSTICS var_diagnostics=DB2_RETURN_STATUS;
--用 GET DIAGNOSTIC 语句获取关于刚刚被执行的 CALL 语句的信息
IF mr.global_var_totalcredit >=50
THEN SET p_new_status ='可授予学士学位';
ELSE SET p_new_status ='学分不够，不足以授予学士学位';
END IF ;
END #
```

（8）调用存储过程 mr.student_status：

```
CALL mr.student_status('马芳',?,?)#
```

执行结果如图 13.7 所示。

```
输出参数的值
------------------------
参数名：P_NEW_STATUS
参数值：可授予学士学位

参数名：VAR_DIAGNOSTICS
参数值：89

返回状态 = 0
```

图 13.7　返回结果

说明

这里故意在存储过程 student_status 中设置了一个 GET DIAGNOSTIC 语句，用于返回刚刚被调用的 CALL 语句状态，实际运用中可删除 GET DIAGNOSTIC 语句和相关参数。

存储过程 student_totalcredit 输出 VAR_DIAGNOSTICS=48；如果该存储过程中没有语句“RETURN global_var_totalcredit”，则输出 VAR_DIAGNOSTICS=0，这里的 0 是 RETURN 语句返回的默认值。

创建多个存储过程共享游标返回的结果集数据，如例 13.6 所示。

DB2 允许外部存储过程使用内部存储过程的一个结果集。步骤如下。

（1）用下面语法声明一个结果集定位器：

```
DECLARE rs_locator_var1 RESULT_SET_LOCATOR VARYING;
```

（2）为被调用存储过程的结果集分配一个结果集定位器：

```
ASSOCIATE RESULT SET LOCATOR(rs_locator_var1) WITH PROCEDURE proc_called;
```

（3）分配一个游标，指向调用存储过程的结果集：

```
ALLOCATE cursor1 CURSOR FOR RESULT SET rs_locator_var1;
```

【例 13.6】 定义一个 SQL 存储过程 result_from_cursor 用于返回学生成绩列表，该列表中包含“学号”“姓名”“课程号”“课程名”“成绩”字段。再定义另一个存储过程 use_nested_cursor，它调用存储过程 result_from_cursor，如果输入某门课程的课程名，则输出该门课程的平均成绩（**实例位置：光盘\TM\sl\13\5**）。



说明

可以在存储过程 result_from_cursor 中用游标返回结果集，该结果集就是上述的“学生成绩列表”中的数据。

事实上，可以用更简捷的存储过程实现“如果输入某门课程的课程名，则输出该门课程的平均成绩”的功能，本例中只是为了讲解多个存储过程如何共享游标返回的结果集数据，而故意将存储过程的定义复杂化。

嵌套的存储过程 use_nested_cursor 调用了存储过程 result_from_cursor 的结果集（结果集没有存储在临时表或永久表中）。

实现步骤如下。

（1）创建存储过程首先需要连接数据库：

```
CONNECT TO mrkj USER mr USING 1#
```

（2）如果已经存在名称为 mr.result_from_cursor 的存储过程，则将其删除：

```
CALL mr.clear_proc('mr','result_from_cursor')#
```

（3）创建存储过程 mr.result_from_cursor：

```
CREATE PROCEDURE mr.result_from_cursor(IN in_cname CHAR(40))
SPECIFIC mr.result_from_cursor
DYNAMIC RESULT SETS 1
P1:BEGIN
DECLARE cursor1 CURSOR WITH RETURN FOR
SELECT sc.学号,s.姓名,sc.课程号,c.课程名,sc.成绩
FROM mr.学生表 s,mr.成绩表 sc,mr.课程表 c
where s.学号=sc.学号 AND c.课程号=sc.课程号 AND c.课程名=in_cname;
```

```
OPEN cursor1;
END P1#
```

（4）调用存储过程 mr.result_from_cursor：

```
CALL mr.result_from_cursor ('计算机应用技术')#
```

本例运行结果如图 13.8 所示。

结果集 1

学号	姓名	课程号	课程名	成绩
14001	马芳	101	计算机应用技术	89
14002	宋静	101	计算机应用技术	83
14003	周天星	101	计算机应用技术	92
14004	蒋阳泉	101	计算机应用技术	84
14005	奎敏	101	计算机应用技术	97

5 条记录已选择。

返回状态 = 0

图 13.8　返回结果

（5）如果已经存在名称为 mr.use_nested_cursor 的存储过程，则将其删除：

```
CALL mr.clear_proc('mr','use _ nested _cursor')#
```

（6）创建存储过程 mr.use_nested_cursor：

```
CREATE PROCEDURE mr.use_nested_cursor (IN in_cname1 CHAR(40),OUT avg_course_score DEC(12,2))
SPECIFIC mr.use_nested_cursor
BEGIN
DECLARE sqlcode,num_records INT DEFAULT 0;
DECLARE var_score,total_course_score DECIMAL(12,2) DEFAULT 0.00;
DECLARE var_sid,var_sname,var_cid,var_cname VARCHAR(40);
DECLARE LOC1 RESULT_SET_LOCATOR VARYING;
SET total_course_score=0.00;
CALL result_from_cursor(in_cname1);
ASSOCIATE RESULT SET LOCATOR(LOC1) WITH PROCEDURE result_from_cursor;
ALLOCATE C1 CURSOR FOR RESULT SET LOC1;
SELECT COUNT(*) INTO num_records
FROM mr.成绩表 sc INNER JOIN mr.课程表 c ON sc.课程号 =c.课程号
where c.课程名=in_cname1;
FETCH FROM c1 into var_sid,var_sname,var_cid,var_cname,var_score;
WHILE sqlcode=0 DO
SET total_course_score=total_course_score+var_score;
FETCH FROM C1 INTO var_sid,var_sname,var_cid,var_cname,var_score;
END WHILE;
SET avg_course_score=total_course_score/num_records;
END #
```

（7）调用存储过程 mr.use_nested_cursor：

```
CALL mr.use_nested_cursor ('计算机应用技术',?)#
```

本例运行结果如图 13.9 所示。

```
---------- 输入的命令 ----------
call mr.use_nested_cursor('计算机应用技术',?)#
--------------------------------------------------------------------------------
call mr.use_nested_cursor('计算机应用技术',?)

输出参数的值
------------------------
参数名：AVG_COURSE_SCORE
参数值：89.00

返回状态 = 0
```

图 13.9　返回结果

创建应用临时表的存储过程，如例 13.7 所示。

在存储过程开发中经常使用临时表。合理地使用临时表可以简化程序的编写，提高执行效率，然而滥用临时表同样也会使得程序运行效率降低。

临时表一般在以下情况下使用：

（1）临时表用于存储程序运行中的临时数据。例如，如果在一个程序中第一条查询语句执行的结果会被后续的查询语句用到，那么，可以把第一次查询的结果存储在一个临时表中供后续查询语句使用，而不是在后续查询语句中重新查询一次。如果第一条查询语句非常复杂且耗时，那么上面的策略是非常有效的。

（2）临时表可以用于存储在一个程序中需要返回多次的结果集。例如，程序中有一个很耗资源的多表查询，同时，该查询在程序中需要执行多次，那么就可以把第一次查询的结果集存储在临时表中，后续的查询只需要访问临时表就可以了。

（3）临时表也可以用于让 SQL 访问非关系型数据库。例如，可以编写程序，把非关系型数据库中的数据插入到一个全局临时表中，那么就可以对其数据进行查询。

可使用 DECLARE GLOBAL TEMPORARY TABLE 语句来定义临时表。DB2 的临时表是基于会话的，且在会话之间是隔离的。当会话结束时，临时表的数据被删除，临时表被隐式卸下。对临时表的定义不会在 SYSCAT.TABLES 中出现。

注意

（1）要在应用程序内创建全局临时表，可使用 DECLARE GLOBAL TEMPORARY TABLE 语句。并且需要创建在用户临时表空间上。

（2）DB2 在数据库创建时，默认情况下并不创建用户临时表空间；如果需要使用临时表，则在第一次创建临时表之前，系统自动创建用户临时表空间（用户临时表空间名称为 SYSTOOLSTMPSPACE）；用户也可以手动创建用户临时表空间（用户可以自定义临时表空间的名称）。

（3）临时表的模式为 SESSION。

（4）默认情况下，在 COMMIT 命令执行时，临时表中的所有记录将被删除；这可以通过创建临时表时指定不同的参数来控制。

（5）运行 ROLLBACK 命令时，用户临时表将被删除。

（6）在 DB2 V8 以上的版本中，可以对临时表记录日志。但可以用 NOT LOGGED 参数指定不记录日志。

【例 13.7】　定义一个存储过程 use_temp_table 用于实现：如果输入某门课程的课程名，则输出该门课程的平均成绩。要求在存储过程 use_temp_table 中应用临时表（**实例位置：光盘\TM\sl\13\6**）。

事实上，本例也可以用更简捷的存储过程实现“如果输入某门课程的课程名则输出该门课程的平均成绩”的功能，本例只是为了讲解临时表在存储过程中的用法，故意将存储过程定义复杂化。

先在存储过程 use_temp_table 中创建一个全局临时表，并将查询获得的结果集数据插入临时表，然后查询临时表。

实现步骤如下。

（1）创建存储过程首先需要连接数据库：

```
CONNECT TO mrkj USER mr USING 1#
```

（2）如果已经存在名称为 mr.use_temp_table 的存储过程，则将其删除：

```
CALL mr.clear_proc('mr','use_temp_table')#
```

（3）创建存储过程 mr. use_temp_table：

```
CREATE PROCEDURE mr. use_temp_table (IN in_cname CHAR(40),OUT avg_course_score DEC(12,2))
SPECIFIC mr. use_temp_table
DYNAMIC RESULT SETS 1
P1:BEGIN
DECLARE GLOBAL TEMPORARY TABLE SESSION.TMP_SC
(
--定义一个全局临时表 TMP_SC
学号 CHAR(6),
姓名 CHAR(16),
课程号 CHAR(6),
课程名 CHAR(40),
成绩 SMALLINT
)WITH REPLACE    --如果存在此临时表，则替换
NOT LOGGED;      --不在日志里记录临时表

--将结果集的数据插入临时表
INSERT INTO SESSION.TMP_SC
SELECT sc.学号,s.姓名,sc.课程号,c.课程名,sc.成绩
FROM mr.学生表 s,mr.成绩表 sc,mr.课程表 c
where s.学号=sc.学号 AND c.课程号=sc.课程号 AND c.课程名=in_cname;

--从临时表中查询某门课程的平均成绩
```

```
SELECT AVG(COALESCE(成绩,0.00)) INTO avg_course_score FROM SESSION.TMP_SC;
END P1#
```

（4）调用存储过程 mr.use_temp_table：

```
CALL mr.use_temp_table ('学前教育',?)#
```

本例运行结果如图 13.10 所示。

```
------------------------------ 输入的命令 ------------------------------
CALL mr.use_temp_table(学前教育',?)#
------------------------------------------------------------------------
CALL mr.use_temp_table(学前教育',?)

输出参数的值
------------------------
参数名： AVG_COURSE_SCORE
参数值： 63.50

返回状态 = 0
```

图 13.10　返回结果

13.1.4　存储过程的重载

DB2 支持存储过程的重载，这些存储过程可以有相同的存储过程名和不同个数的参数，只要给它们指定不同的“特定名称”（或者用系统自动产生的默认特定名称）即可。

注意

DB2 允许存在同名但参数数目不同的存储过程；但不允许存在同名且参数数目相同的存储过程，即使参数数据类型不同也不行。

【例 13.8】　定义一个存储过程 proc_with_parameter 用于实现下列功能：如果输入某个学生的学号，则将该学生的姓名改为某某（**实例位置：光盘\TM\sl\13\7**）。

实现步骤如下。

（1）创建存储过程首先需要连接数据库：

```
CONNECT TO mrkj USER mr USING 1#
```

（2）如果已经存在名称为 mr.proc_with_parameter 的存储过程，则将其删除：

```
CALL mr.clear_proc('mr','proc_with_parameter')#
```

（3）创建存储过程 mr. use_temp_table：

```
CREATE PROCEDURE mr.proc_with_parameter (IN in_sid CHAR(6))
SPECIFIC mr.proc_with_parameter1
LANGUAGE SQL UPDATE mr.学生表 SET 姓名='某某'
WHERE 学号 =in_sid#
```

（4）调用存储过程 mr.proc_with_parameter：

```
CALL  mr.proc_with_parameter ('14003')#
```

如果存在同名的存储过程，如何将其中的一个删除呢？

这时需要根据这些存储过程的特定名称来删除它们。例如，可以用下面的语句删除新创建的存储过程“mr.proc_with_parameter(IN in_sid CHAR(6))”：

```
DROP SPECIFIC PROCEDURE mr.proc_with_parameter1#
```

此时已不存在名称同为 proc_with_parameter 的存储过程，可用下面的语句删除存储过程 proc_with_parameter(CHAR(6),CHAR(16))：

```
DROP SPECIFIC PROCEDURE mr.proc_with_parameter#
```

或

```
DROP PROCEDURE mr.proc_with_parameter#
```

或

```
DROP PROCEDURE mr.proc_with_parameter(CHAR(6),CHAR(16))#
```

13.1.5　存储过程的查询、修改和删除

1. 查询

存储过程的所有信息都记录在 SYSCAT.PROCEDURES 系统目录视图中。可用下面的命令查询已经创建的用户存储过程。

查询用户存储过程的语法格式为：

```
SELECT * FROM SYSCAT.PROCEDURES
[WHERE PROCSCHEMA='USER_DEFINED_PROCEDURE_SCHEMA'
[AND PROCNAME='USER_DEFINED_PROCEDURE_NAME']
[AND SPECIFICNAME='USER_DEFINED_PROCEDURE_SPECIFIC_NAME']]
```

说明

（1）USER_DEFINED_PROCEDURE_SCHEMA：SQL 存储过程的模式名（必须大写）。

（2）USER_DEFINED_PROCEDURE_NAME：SQL 存储过程的名称（必须大写）。

（3）USER_DEFINED_PROCEDURE_SPECIFIC_NAME：SQL 存储过程的特定名称（必须大写）。

2. 修改

DB2 存储过程一旦创建就不能够修改，只能删除。如果需要修改存储过程，必须先删除原来的存储过程，然后再创建新的存储过程。

3. 删除

当不再使用一个存储过程时，就要把它从数据库中删除。使用 DROP PROCEDURE 语句可以永久删除存储过程。在删除之前，必须确认存储过程没有任何依赖关系。

删除存储过程的语法格式为：

```
DROP PROCEDURE procedure_name#
```

或

```
DROP PROCEDURE procedure_name()#
```

或

```
DROP PROCEDURE procedure_name (data_type[,...n])#
```

或

```
DROP SPECIFIC PROCEDURE procedure_specific_name #
```

注意

如果存在同名的但特定名称不同的存储过程，必须使用 DROP SPECIFIC PROCEDURE 语句才能删除其中一个存储过程。

【例 13.9】 删除 MRKJ 数据库中的 MEDIAN_RESULT_SET 存储过程（**实例位置：光盘\TM\sl\13\8**）。

实现步骤如下。

（1）创建存储过程首先需要连接数据库：

```
CONNECT TO mrkj USER mr USING 1#
```

（2）删除存储过程：

```
DROP PROCEDURE mr.median_result_set#
```

也可以在“控制中心”对象视图中选择 MEDIAN_RESULT_SET 存储过程，然后再单击鼠标右键，在弹出的快捷菜单中选择“删除”命令，从而删除此存储过程。

13.2 触 发 器

触发器可以看作是一种“特殊”的存储过程，它定义了一些与数据库相关的事件发生时应执行的“功能代码块”，通常用于管理复杂的完整性约束，或监控对表的修改，或通知其他程序，甚至可以实现对数据库的审计功能。

13.2.1　触发器的概念

触发器定义一组操作，这组操作与用于指定表或类型表的 INSERT、UPDATE 或 DELETE 子句一起执行，或由这些子句触发。

触发器有以下用途：

（1）验证输入数据。例如，MRKJ 数据库有“学生表”、“成绩表”和“课程表”，当向“成绩表”插入某一学号学生的某一课程成绩时，该“学号”应是“学生表”中已存在的，此时，可通过定义 INSERT 触发器实现上述功能。

（2）利用触发器可以方便地实现数据库中数据的完整性。可将触发器与引用约束和检查约束配合使用，以强制执行数据完整性规则。可通过触发器实现多个表之间数据的一致性。

（3）利用触发器可使公司的处理任务自动进行。例如，在销售系统中，通过更新触发器，可以检测什么时候库存下降到了需要再进货的量，并自动生成给供货商的订单。

（4）可使用触发器支持一般形式的完整性或业务规则。例如，在接受订单或更新总结数据表之前，触发器可以检查客户的信用额度。

触发器具有以下优点：

（1）更快地开发应用程序。因为触发器存储在数据库中，所以不必编写触发器在每个应用程序中执行的操作。

（2）更容易维护。一旦定义了一个触发器，那么，当访问创建它所基于的表时，会自动调用该触发器。

（3）业务规则的全局实现。如果业务策略改变，只需要更改触发器而不必更改每个应用程序。

同一个表上的多个触发器的执行顺序为：可以对表、事件（INSERT、UPDATE、DELETE 和 INSTEAD OF）或激活事件（BEFORE 和 AFTER）的组合指定多个触发器，即使同一类型的触发器，也可使用多个。当对特定表、事件和激活事件存在多个触发器时，激活触发器的顺序与创建它们的顺序相同。因此，最后创建的触发器就是最后激活的触发器。

触发器级联：激活触发器可能导致触发器级联，这是由于激活了一个执行语句的触发器，这些语句导致激活其他触发器或再次激活相同触发器。触发操作还可能导致应用完整性规则引发的更新，从而可能导致激活更多触发器。借助触发器级联，可能会激活触发器或完整性规则链，从而由于单个 INSERT、UPDATE 或 DELETE 语句而导致对数据库的大幅度更改。

触发器冲突的解决：当多个触发器对同一个对象执行插入、更新或删除操作时，会使用冲突解决机制来访问冲突。这样可能会对性能产生显著影响，在分区数据库环境中尤其如此。

13.2.2　触发器的类型

触发器主要包括前触发器、后触发器和 INSTEAD OF 触发器等 3 种，下面分别进行讲解。

1. 前触发器

前触发器是在更新、插入或删除操作前运行。

如果是 BEFORE UPDATE 或 BEFORE INSERT 触发器，可以在实际修改数据库之前修改要更新或插入的值；这些触发器可用来在需要时将应用程序中的输入（数据的用户视图）变换为内部数据库格式，还可以运行用户定义的函数中编写的其他非数据库操作。如果是 BEFORE DELETE 触发器，那么它在删除操作前运行，用于检查值（必要时产生错误）。

2. 后触发器

后触发器是在更新、插入或删除操作后运行。

更新或插入操作后运行的触发器有下列几种用途：

（1）更新其他表中的数据。此功能对于保持数据之间的关系或保留审计跟踪信息很有用。

（2）针对表或其他表中的其他数据进行检查。当引用完整性约束不适合或者表检查约束限制只对当前表进行检查时，此功能对于确保数据完整性很有用。

（3）运行用户定义的函数中编写的非数据库操作。在发出警报或更新数据库外的信息时，此功能很有用。

3. INSTEAD OF 触发器

描述如何对视图执行插入、更新和删除操作，这些视图太复杂，以致无法在本机支持这些操作。这种触发器允许应用程序将视图用作所有 SQL 操作（插入、删除、更新和选择）的唯一界面。

13.2.3 用 CREATE TRIGGER 语句创建触发器

1. CREATE TRIGGER 语句

CREATE TRIGGER 语句的基本结构为：

```
CREATE TRIGGER <tigger_name>
<action> ON <table_name | view_name>
<operation>
<triggered_action>
```

语句的语法为：

```
CREATE TRIGGER tigger_name [NO CASCADE]
[BEFORE | AFTER | INSTEAD OF]
[INSERT | DELETE | UPDATE [OF column_name]]
ON <table_name | view_name>
REFERENCING
[OLD[AS]correlation_name | NEW [AS] correlation_name | OLD TABLE [AS] identifier
| NEW TABLE [AS] correlation_name]
[FOR EACH <ROW | STATEMENT>]
triggered_action
```

其中：

```
triggered_action ::=
[WHEN (search_condition)]
[label_name :]SQL_procedure_statement
SQL_procedure_statement ::=
SQL_routine_statement | dynamic_compound_statement
```

参数说明如下。

- ☑ NO CASCADE：指定了触发器的触发动作不会激活其他触发器。
- ☑ BEFORE：在对受影响的表的任何实际更新落实到数据库前执行相关的触发器动作。
- ☑ AFTER：在对受影响的表的任何实际更新落实到数据库后执行相关的触发操作。
- ☑ INSTEAD OF：指定触发器动作代替了对应的受影响的视图的操作。只有 INSTEAD OF 触发器允许执行对给定受影响的视图的每一种操作。
- ☑ REFERENCING：指定转换变量的相关名和转换表的表名。触发器执行前和执行后数据库的状态不同，有两类方法可以标识数据库的不同状态，并且可引用受触发器影响的数据信息。
  - ➢ 第一类方法是把即将或已经受触发器影响的行集合中的一行定义成“转换变量”，并且给转换变量定义一个相关名。
    - OLD AS correlation_name：当前触发器执行前，受影响的行集合中的一行的相关名为 correlation_name。每当引用当前触发器执行前的某一行中的某一列的值时，就在列名前加上 correation_name。
    - NEW AS correlation_name：当前触发器或者 BEFORE 触发器中的 SET 语句执行后，受影响的行集合中的一行的相关名为 correlation_name。每当引用当前触发器或者 BEFORE 触发器执行后的某一行中的某一列的值时，就在列名前加上 correlation_name。
  - ➢ 第二类方法是把触发器执行前将受影响或触发器执行后已经受影响的行集合看成一个“转换表”，并且给转换表定义一个表名。
    - OLD TABLE AS identifier：将当前触发器执行前即将受影响的行的集合组成转换表，转换表的表名为 identifier。
    - NEW TABLE AS identifier：当前触发器或 BEFORE 触发器中的任何 SET 语句执行后，已经受影响的行的集合组成转换表，转换表的表名为 identifier。
- ☑ FOR EACH ROW：被触发的 SQL 语句的动作应用到受影响的表或视图的每一行。
- ☑ FOR EACH STATEMENT：被触发的 SQL 语句集合中的每一个 SQL 语句的动作只执行一次。
- ☑ triggered_action：定义触发器被激活时执行的动作，触发器动作由一个 SQL 存储过程和一个可选的搜索条件语句组成，满足该搜索条件时，执行 SQL 存储过程语句。
- ☑ SQL_procedure_statement：SQL 存储过程语句。

以上的语法中需要注意的是，OLD | NEW | OLD TABLE| NEW TABLE 在使用时，需要遵循以下规则：

- ☑ OLD 和 NEW 在每个触发器中只能指定一次。
- ☑ OLD TABLE 和 NEW TABLE 在每个触发器中只能指定一次，而且只能指定给 AFTER 触发器或 INSTEAD OF 触发器。

☑ DELETE 触发器中不允许 NEW 相关名和 NEW_TABLE 名。

☑ INSERT 触发器中不允许 OLD 相关名和 OLD_TABLE 名。

☑ BEFORE 触发器中不允许 OLD_TABLE 名和 NEW_TABLE 名。

2. 举例

设计一个存储过程，用来删除已有的指定名称的用户触发器，如例 13.10 所示。

【例 13.10】 设计一个存储过程，用于查询系统中是否已经存在用户定义的名称为 tigger_id 的触发器，如果不存在该触发器，则不做任何操作，如果存在该触发器，则将其删除（**实例位置：光盘\TM\sl\13\9**）。

分析：

在本书后面章节定义其他示例触发器前，都调用了这里定义的 clear_tigger 存储过程。

触发器的所有定义都记录在 SYSCAT.TRIGGERS 系统目录视图中，如果该系统目录视图中存在名称为 tigger_in 的触发器的信息，则表示已经存在名称为 tigger_in 的触发器，需用 DROP TRIGGER 语句删除该触发器。

实现步骤如下所示。

（1）连接到 MRKJ 数据库：

```
CONNECT TO mrkj USER mr USING 1;
```

（2）创建存储过程 clear_tigger：

```
CREATE PROCEDURE mr.clear_trigger
(IN trigger_schema VARCHAR(128), IN trigger_in VARCHAR(128))
SPECIFIC mr.clear_trigger
BEGIN ATOMIC
DECLARE trigger_count INTEGER default 0;
DECLARE statement VARCHAR(200);
SELECT COUNT(T.TRIGNAME) INTO trigger_count FROM SYSCAT.TRIGGERS T
WHERE T.TRIGNAME=UPPER(trigger_in)
AND T.TRIGSCHEMA=UPPER(trigger_schema);
IF (trigger_count=1)
THEN SET statement=
'DROP TRIGGER'||UPPER(trigger_schema)||'.'||upper(trigger_in);
EXECUTE IMMEDIATE statement;
RETURN 1;
END IF;
END#
```

【例 13.11】 创建一个触发器，它在每次向“成绩表”增加一条记录且该记录中的成绩字段大于等于 60 分时，就将“学生表”中相应学生的“总成绩”更新为当前“总成绩”的值加上相应课程的学分数（**实例位置：光盘\TM\sl\13\10**）。

实现步骤如下。

（1）创建存储过程首先需要连接数据库：

```
CONNECT TO mrkj USER mr USING 1#
```

（2）如果已经存在名称为 mr.addrecord_cjb 的存储过程，则将其删除：

```
CALL mr.clear_trigger('mr','addrecord_cjb')#
```

（3）创建触发器 mr.addrecord_cjb：

```
CREATE TRIGGER mr.addrecord_cjb
AFTER INSERT ON mr.成绩表        --后触发器
REFERENCING NEW AS N             --将触发器执行后受影响的行定义为 N
FOR EACH ROW
WHEN (60<=(SELECT  成绩 FROM mr.成绩表 WHERE 学号 =N.学号 AND 课程号=N.课程号))
--已经插入的记录中成绩大于等于 60 分
BEGIN ATOMIC
DECLARE credit SMALLINT DEFAULT 0;
--credit 表示刚刚插入的记录中课程对应的学分
IF(0<(SELECT c.学分 FROM mr.课程表 c WHERE c.课程号=N.课程号))
THEN UPDATE mr.学生表 s SET 总分数 =总分数+credit
WHERE s.学号=N.学号;
END IF ;
END#
```

使用触发器定义业务规则的示例，如例 13.12 所示。

【例 13.12】　假设学校规定，凡是修够 50 个学分的学生都可以获得学士学位，修够 50 个学分的学生名单必须进入“学位表”，但“学位表”中并不只包含“学生表”中修够 50 个学分的学生名单（**实例位置：光盘\TM\sl\13\11**）。

“学位表”创建命令如下：

```
CONNECT TO mrkj USER mr USING 1#
CREATE TABLE mr.学位表
(
    学号 CHARACTER(8) NOT null,
    姓名 CHARACTER(20),
    专业 CHARACTER(40),
    授予时间 DATE,
    学位类型 CHARACTER(20),
    授予状态 CHARACTER(40)
)IN CUSTOMDATASPACE1#
```

说明

这种业务规则只能用触发器来表示，不能用在表上声明外键约束来表示，因为“学生表”中的学生名单并不全在“学位表”中，“学位表”中的名单也不全在“学生表”中。

实现步骤如下。

（1）需要先连接数据库：

```
CONNECT TO mrkj USER mr USING 1#
```

（2）如果已经存在名称为 mr.student_to_degree 的触发器，则将其删除：

```
CALL mr.clear_trigger('mr','student_to_degree')#
```

（3）创建触发器 mr.student_to_degree：

```
CREATE TRIGGER mr.student_to_degree
      AFTER UPDATE ON mr.学生表
      REFERENCING NEW AS N
      FOR EACH ROW
WHEN (N.总分数 >=50 AND N.学号 NOT IN (SELECT s.学号 FROM mr.学生表 s))
BEGIN ATOMIC
 INSERT INTO mr.学位表(学号,姓名,专业,学位类型,授予状态)
VALUES(N.学号,N.姓名,N.专业,'工学学士','可以授予但还未授予');
END #
```

使用触发器验证输入数据，从而实现数据库中数据的完整性。如下面的例 13.13 所示。

【例 13.13】 在数据库 MKRJ 中创建一个触发器，当向“成绩表”中插入一条记录时，检查该记录的“学号”在“学生表”中是否存在，检查“课程号”在“课程表”中是否存在，若有一项为否，则不允许插入。可使用两种方法来实现此目的（**实例位置：光盘\TM\sl\13\12**）。

定义两个外键约束，它强制使“成绩表”中的“学号”字段成为“学生表”中“学生”字段的值，强制使“成绩表”中的“课程号”字段成为“课程表”中“课程号”字段的值。

```
ALTER TABLE mr.成绩表
ADD CONSTRAINT cjb_foreign_key1
FOREIGN KEY (学号) REFERENCES mr.学生表(学号)
ON DELETE CASCADE
ON UPDATE RESTRICT
ENFORCED
ENABLE QUERY OPTIMIZATION
ADD CONSTRAINT cjb_foreign_key2
foreign key(课程号) REFERENCES mr.课程表(课程号)
ON DELETE CASCADE
ON UPDATE RESTRICT
ENFORCED
ENABLE QUERY OPTIMIZATION #
```

按照如下步骤定义一个前触发器。

（1）需要先连接数据库：

```
CONNECT TO mrkj USER mr USING 1#
```

（2）如果已经存在名称为 mr.block_insert_cjb 的触发器，则将其删除：

```
CALL mr.clear_trigger('mr','block_insert_cjb')#
```

（3）创建触发器 mr.block_insert_cjb：

```
CREATE TRIGGER mr.block_insert_cjb
BEFORE INSERT ON mr.成绩表
```

```
REFERENCING NEW AS N
FOR EACH ROW
BEGIN ATOMIC
IF(N.学号 NOT IN(SELECT s.学号 FROM mr.学生表 s))
THEN SIGNAL SQLSTATE'70001' ('错误:该记录的学号在学生表中不存在!');
END IF;
IF(N.课程号 NOT IN (SELECT c.课程号 FROM mr.课程表 c))
THEN SIGNAL SQLSTATE'70002'('错误:该记录的课程号在课程表中不存在!');
END IF;
END #
```

先创建两个外键约束 cjb_foreign_key1 和 cjb_foreign_key2，然后再创建前触发器 block_insert_cjb，最后在“命令编辑器”中运行下面的语句：

```
INSERT INTO mr.成绩表 VALUES('14567','201',90)#
```

可以发现，“成绩表”中并没有插入这条记录，而且系统返回错误信息，如图 13.11 所示。

```
DB21034E 该命令被当作 SQL 语句来处理，因为它是无效的"命令行处理器"命令。在
SQL 处理期间，它返回：
SQL0438N 应用程序发生错误或警告，其诊断文本为："错误:该记录的学号在学生表中不存
在!"。 SQLSTATE=70001

SQL0438N 应用程序发生错误或警告，其诊断文本为："错误:该记录的学号在学生表中不存在!"。
```

图 13.11　返回错误信息

以上操作证明外键约束 cjb_foreign_key1 和 cjb_foreign_key2 还没有执行，也就是说，前触发器 block_insert_cjb 是先于外键约束 cjb_foreign_key1 和 cjb_foreign_key2 执行的。

如例 13.14 所示进行 INSTEAD OF 触发器的设计。

如果视图的数据来自于多个基表，则必须使用 INSTEAD OF 触发器支持引用表中数据的插入、更新和删除操作。

通过使用 INSTEAD OF 触发器，请求对视图执行的修改操作将替换为触发器逻辑，该逻辑代表视图执行操作。从应用程序的角度看，这是透明地进行的，因为它看到所有操作都是对视图执行的。只允许将一个 INSTEAD OF 触发器用于给定主题视图的每种操作。

若在一个多表视图上定义了 INSTEAD OF INSERT 触发器，视图各列的值可能允许为空，也可能不允许为空，若视图某列的值不允许为空，则 INSERT 语句必须为该列提供相应的值。

视图本身必须是隐式类型视图或解析为隐式类型视图的别名。此外，它不能是使用 WITH CHECK OPTION 定义的视图（使用检查选项的视图也称为“对称视图”）或在其上直接或间接定义了“对称视图”的视图。

【例 13.14】 创建 3 个 INSTEAD OF 触发器，它们分别为已定义的视图（“图书视图”）提供 INSERT、UPDATE 和 DELETE 逻辑（**实例位置：光盘\TM\sl\13\13**）。

（1）使用下面的语句定义“图书信息表”、“图书报价表”和“图书视图”：

```
CONNECT TO mrkj USER mr USING 1#
--定义“图书信息表”
```

```
CREATE TABLE mr.图书信息表
(
      书号 CHARCHAR(20),
      书名 VARCHAR(80),
      作者 VARCHAR(20)
)#

--定义“图书报价表”

CREATE TABLE mr.图书报价表
(
      书号 CHARACTER(20),
      价格 DECIMAL(10,2)
)#

--定义“图书视图”

CREATE VIEW mr.图书视图(书号,书名,价格)
AS
      SELECT bx.书号,书名,价格
FROM mr.图书信息表 bx,mr.图书报价表 bj
WHERE bx.书号=bj.书号#
```



说明

视图“图书视图”的 FROM 子句中包含连接，因此不能在本机支持任何修改（包括插入、更新、删除）操作，除非定义了 INSTEAD OF 触发器。

（2）INSTEAD OF 触发器代替对视图的插入操作。

在“图书视图”上定义一个 INSTEAD OF 触发器，该 INSTEAD OF 触发器代替对“图书视图”的插入操作，如果 MRKJ 数据库中没有下面的触发器，系统就不允许用户对“图书视图”执行插入操作。

实现步骤如下。

① 先需要连接数据库：

```
CONNECT TO mrkj USER mr USING 1#
```

② 如果已经存在名称为 mr.bookview_insert 的触发器，则将其删除：

```
CALL mr.clear_trigger('mr','bookview_insert')#
```

③ 创建触发器 mr.bookview_insert：

```
CREATE   TRIGGER mr. bookview_insert
INSTEAD OF INSERT ON mr.图书视图
REFERENCING NEW AS N_bp
FOR EACH ROW
INSERT INTO mr.图书报价表(书号,价格)
VALUES(COALESCE((SELECT 书号 FROM mr.图书信息表 AS bx
```

```
WHERE bx.书号=N_bp.书号),RAISE_ERROR('70003','不存在的书号')),
COALESCE(价格,0.00))#
```

④ 测试上面定义的 INSTEAD OF 触发器。

如果这时“图书信息表”中不存在书号为“ISBN978-7-302-289”的图书信息，那么此时执行下面的命令就会报错：

```
INSERT INTO mr.图书视图
VALUES('ISBN978-7-302-289','书名',59.8)#
```

返回错误信息如图 13.12 所示。

```
SQL处理期间，它返回：
SQL0438N 应用程序发生错误或警告，其诊断文本为："不存在的书号"。
SQLSTATE=70003

SQL0438N 应用程序发生错误或警告，其诊断文本为："不存在的书号"。
```

图 13.12　返回错误信息

如果先执行下面的命令，然后再次执行上面的“INSERT INTO 图书视图”语句，这时就能够在“图书视图”视图和“图书报价表”中分别正确地插入一条记录。

```
INSERT INTO mr.图书信息表
VALUES('ISBN978-7-302-289','书名',59.8)#
```

（3）INSTEAD OF 触发器代替对视图的更新操作。

创建第 2 个 INSTEAD OF 触发器，该 INSTEAD OF 触发器代替对视图“图书视图”的更新操作，如果 MRKJ 数据库中没有下面的触发器，系统就不允许用户对“图书视图”执行更新操作。

实现步骤如下。

① 需要先连接数据库：

```
CONNECT TO mrkj USER mr USING 1#
```

② 如果已经存在名称为 mr.bookview_update 的触发器，则将其删除：

```
CALL mr.clear_trigger('mr','bookview_update')#
```

③ 创建触发器 mr.bookview_ update：

```
CREATE   TRIGGER mr.bookview_update
INSTEAD OF UPDATE ON mr.图书视图
REFERENCING NEW AS N_bp   OLD AS O_bp
FOR EACH ROW
BEGIN ATOMIC
VALUES(
CASE
WHEN N_bp.书号=O_bp.书号  THEN 0
ELSE RAISE_ERROR('70004','不能修改书号')
END);
```

```
UPDATE mr.图书报价表 AS bp
SET (书号,价格)=
(COALESCE((SELECT 书号 FROM mr.图书信息表 AS bi
WHERE bi.书号=N_bp.书号),
RAISE_ERROR('70003','不存在的书号')),
COALESCE(N_bp.价格,0.00))
WHERE N_bp.书号=bp.书号;
END #
```

④ 测试上面定义的 INSTEAD OF 触发器。

如果这时执行下面命令，从而在“图书信息”上执行 UPDATE 操作，系统就会报错：

```
UPDATE INTO mr.图书视图
SET 书号='ISBN978-7-302-289' WHERE 价格=59.8#
```

（4）INSTEAD OF 触发器代替对视图的删除操作。

创建第 3 个 INSTEAD OF 触发器，该 INSTEAD OF 触发器代替对视图“图书视图”的删除操作，如果 MRKJ 数据库中没有下面的触发器，系统就不允许用户对“图书视图”执行删除操作。

实现步骤如下。

① 先需要连接数据库：

```
CONNECT TO mrkj USER mr USING 1#
```

② 如果已经存在名称为 mr.bookview_delete 的触发器，则将其删除：

```
CALL mr.clear_trigger('mr','bookview_delete')#
```

③ 创建触发器 mr.bookview_delete：

```
CREATE  TRIGGER mr.bookview_delete
INSTEAD OF DELETE ON mr.图书视图
REFERENCING OLD AS O_bp FOR EACH ROW
DELETE FROM mr.图书报价表 AS bp WHERE bp.书号= O_bp.书号#
```

④ 测试上面定义的 INSTEAD OF 触发器：

如果这时在“命令编辑器”中运行下面的语句就可以同时删除“图书报价表”和“图书视图”中书号为“ISBN978-7-302-289”的相关记录，但会保留“图书信息表”中书号为“ISBN978-7-302-289”的相关记录。

```
DELETE  FROM mr.图书视图 WHERE 书号='ISBN978-7-302-289'#
```

13.2.4 查询、修改和删除触发器

1. 查询

触发器的所有定义都记录在 SYSCAT.TRIGGERS 系统目录视图中。可用下面的语句查询用户创建

的所有触发器的信息：

```
SELECT * FROM syscat.triggers where trigschema='MR'#
```

查询结果如图 13.13 所示。

TRIGSCHEMA	TRIGNAME	OWNER	OWNERTYPE	TABSCHEMA	TABNAME	TRIGTIME	TRIGEV
MR	ADDRECORD_...	MR	U	MR	成绩表	A	I
MR	BLOCK_INSER...	MR	U	MR	成绩表	B	I
MR	BOOKVIEW_DE...	MR	U	MR	图书视图	I	D
MR	BOOKVIEW_INS...	MR	U	MR	图书视图	I	I
MR	BOOKVIEW_UP...	MR	U	MR	图书视图	I	U
MR	STUDENT_TO_...	MR	U	MR	学生表	A	U

图 13.13　查询所有触发器

触发器与其他对象的所有依赖性都记录在 SYSCAT.TRIGGERS 系统目录视图中。可用下面的语句查询创建的所有用户触发器对表或视图的依赖关系：

```
SELECT * FROM syscat.trigdep where trigschema='MR'#
```

查询结果如图 13.14 所示。

TRIGSCHEMA	TRIGNAME	BTYPE	BSCHEMA	BMODULENAME	BNAME	BMODULEID	TABA
MR	ADDRECORD_...	T	MR		学生表		
MR	ADDRECORD_...	T	MR		成绩表		
MR	ADDRECORD_...	T	MR		课程表		
MR	BLOCK_INSER...	T	MR		学生表		
MR	BLOCK_INSER...	T	MR		成绩表		
MR	BLOCK_INSER...	T	MR		课程表		

图 13.14　查询依赖关系

注意

MR 是前面已经创建的那些用户存储过程的模式名（这里的模式名必须大写），可以根据具体情况更改语句中的这个变量。

2. 修改

只能修改触发器的注释，触发器的其他要素都不能修改。如果要修改注释以外的其他要素必须要删除触发器，然后根据需要的新定义再次创建它们。

【例 13.15】 修改前面已经创建的触发器 ADDRECORD_CJB 的注释。

在“控制中心”中修改触发器注释的实现步骤如下：

（1）在“控制中心”对象视图中选择 MRKJ→“触发器”文件夹，在内容窗格中的触发器列表中选择触发器 ADDRECORD_CJB，如图 13.15 所示，然后单击鼠标右键，在弹出的快捷菜单中选择“改变”命令，弹出“改变触发器”对话框。



图 13.15 选择触发器

（2）在“改变触发器”对话框中的“注释”文本框中输入“每次向成绩表增加一条记录且该记录中的成绩字段大于等于 60 分时，就将学生表中相应学生的总分数更新为当前总分数的值加上相应课程的学分数”，如图 13.16 所示。单击“显示 SQL”按钮，可以查看 SQL 脚本，如图 13.17 所示。单击“确定”按钮，完成对触发器 ADDRECORD_CJB 注释的修改。

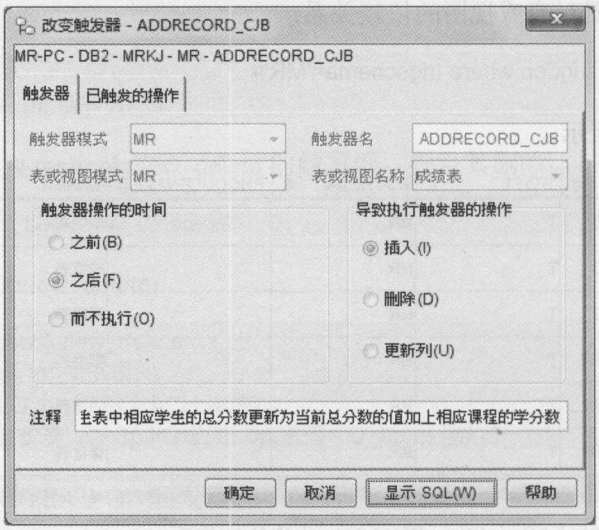


图 13.16 修改注释

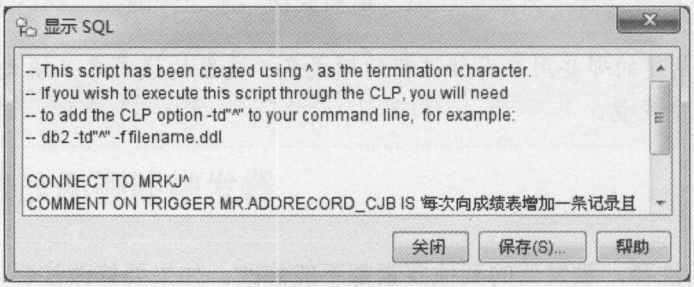


图 13.17 查看 SQL 命令

使用下面的 SQL 语句来修改触发器 ADDRECORD_CJB 的注释。

实现步骤如下。

（1）需要先连接数据库：

```
CONNECT TO mrkj USER mr USING 1#
```

（2）修改触发器 ADDRECORD_CJB 的注释：

```
COMMENT ON TRIGGER mr.addrecord_cjb IS '每次向成绩表增加一条记录且该记录中的成绩字段大于等于 60 分时，就将学生表中相应学生的总分数更新为当前总分数的值加上相应课程的学分数'#
```

3. 删除

一个触发器可以依赖许多个对象。如果触发器所依赖的某个对象被删除，那么该触发器就不可用，但它的定义仍然保留在系统目录视图中。如果删除触发器，那么它的描述会从 SYSCAT.TRIGGERS 系统目录视图中被删除，且它的所有依赖性也会从 SYSCAT.TRIGGERS 系统目录视图中被删除。所有与该触发器有 INSERT、UPDATE、DELETE 关系的程序包都变成无效。

在“控制中心”对象视图中选择 MRKJ→“触发器”文件夹，在内容窗格中的触发器列表中选择触发器“ADDRECORD_CJB”，然后单击鼠标右键，在弹出的快捷菜单中选择“删除”命令，如图 13.18 所示，在弹出的对话框中单击“确定”按钮。

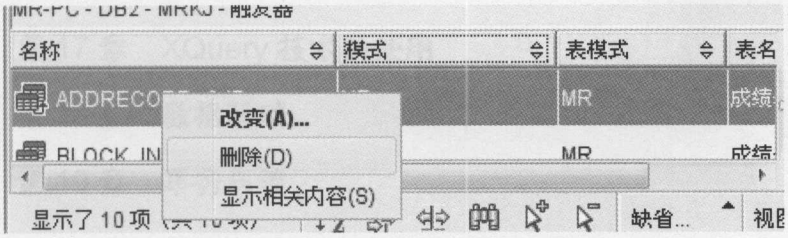


图 13.18　删除触发器

删除触发器的语法如下：

```
DROP TRIGGER trigger_name#
```

参数说明如下。

trigger_name：指要删除的触发器名称，包含触发器的模式名。

【例 13.16】　删除触发器 ADDRECORD_CJB（**实例位置：光盘\TM\sl\13\14**）。

实现步骤如下。

（1）需要先连接数据库：

```
CONNECT TO mrkj USER mr USING 1#
```

（2）删除 ADDRECORD_CJB 触发器：

```
DROP TRIGGER mr.addrecord_cjb#
```

13.3 小　　结

本章着重介绍了存储过程和触发器。学习本章内容时，应该重点掌握存储过程和触发器的使用，因为它们在应用系统开发中经常被使用。

13.4 实践与练习

（1）什么是存储过程？简述存储过程的作用与分类。（**答案位置：光盘\TM\sl\13\15**）

（2）触发器分为哪 3 类，分别具有哪些用途？（**答案位置：光盘\TM\sl\13\16**）



高级应用

- 第 14 章　DB2 数据库安全处理
- 第 15 章　数据库的备份与恢复
- 第 16 章　PureXML 技术应用
- 第 17 章　XQuery 技术的使用
- 第 18 章　数据移动
- 第 19 章　事务与锁

本篇介绍 DB2 数据库安全处理、数据库的备份与恢复、PureXML 技术应用、XQuery 技术的使用、数据移动、事务与锁等。学习完这一部分，能够实现对 DB2 数据库的安全处理、备份和恢复数据库、使用 DB2 数据库导入、导出数据等操作。

第14章

DB2 数据库安全处理

（视频讲解：72 分钟）

本章主要介绍如何进行系统安全处理，主要包括安全的概述、认证机制、身份验证和授权、DB2 身份验证、管理权限、特权、LBAC 凭证、GET AUTHORIZATIONS 命令和使用模式控制对数据库对象的访问等。讲解过程中为了便于读者理解，结合了相应的例子。读者可以熟练掌握系统安全的一些用法，并应用权限编写 SQL 语句，从而分配用户访问权限和提高系统安全。

通过阅读本章，您可以：

- 了解有关数据库安全的基础知识
- 了解认证机制的过程
- 了解身份验证和授权
- 掌握 DB2 数据库的身份验证
- 掌握如何管理权限
- 熟悉 DB2 数据库的特权
- 了解 LBAC 凭证
- 熟悉 GET AUTHORIZATIONS 命令
- 使用模式控制对数据库对象的访问

14.1　DB2 数据库安全概述

数据是企业的生命线，特别是银行、保险、通信等一些行业，数据库中保存了大量的客户信息，假如这些关键信息被某些不法分子获取，并对数据进行修改、删除、复制、传播等，将给企业和个人造成巨大损失。

数据安全威胁不仅来自外部，也可能来自内部，正所谓日防、夜防、家贼难防。在国内某些企业，安全意识比较薄弱，安全机制还不够健全，某些人的权限很大，假如没有机制记录他们的操作行为，而仅凭自觉和道德约束，难免会出现重要数据被更改的情况（例如在自己的存款账户上加几个 0）。制定严密的可操作的安全策略是每个企业必须严肃对待的一件事情。

在安全领域，有个 4A 理论：Authentication（认证）、Authorization（授权）、Account（用户/账户）、Audit（审计）。认证是第一屏障，通过验证用户身份，判断是否可以通过；权限是指用户认证通过后，能够执行什么操作，例如有的用户有读取权限，有的用户可以增、删、改；审计类似飞机的黑匣子，能够记录一些重要的操作，是内控的有效手段（但是审计永远都是事后审计，只能追溯到问题的原因）。

14.2　认 证 机 制

认证是验证用户身份的过程，可以把数据库想象成为一个房子，认证（身份信息）就是房子的钥匙，只有通过认证，才能进入房子，否则就被挡在外面。DB2 支持认证的形式为：操作系统用户名/密码认证、Kerberos 认证和第三方安全插件。

图 14.1 是当客户端发出 connect to mrkj user mr using 1 命令时，服务器端进行认证的过程：

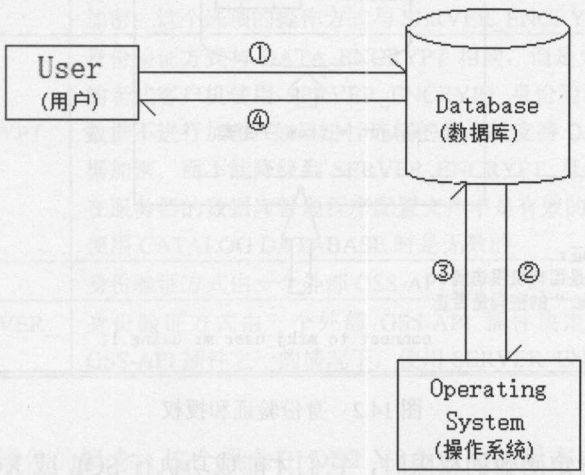


图 14.1　DB2 认证过程

（1）在数据库连接时提供用户名和密码，这些信息会提交给数据库服务器。

（2）数据库服务器将用户名和密码传递给安全插件，安全插件调用操作系统 API 验证用户名和密码是否正确。

（3）操作系统验证用户名和密码，并将结果返回给安全插件，安全插件再返回给数据库服务器。

（4）如果安全检查返回的信息表明用户名和密码没有通过操作系统认证，则返回错误信息给连接用户。如果用户名和密码正确，将会加载这个用户所属的组信息，并通过 syscat.dbauth 验证该用户/组/PUBLIC 是否有连接权限，如果有，则允许连接；如果没有，则不允许连接。

14.3 身份验证和授权

安装和操作 DB2 UDB 需要某些用户和组账号。要记住的是：这些账号是在 DB2 UDB 外的安全设施中定义的——通常是操作系统。用户账号管理（包括密码策略、命名约定和组定义）都是在这个外部设置上进行的。

当提交 DB2 请求时，DB2 UDB 与这个外部设置进行交互，从而检验提供的用户 ID 和密码。这种交互称为“身份验证”，显示在图 14.2 的下半部分。DB2 UDB 依赖于这个外部安全设置对用户进行身份验证，只有身份验证成功时才授予对 DB2 UDB 资源的访问权。

身份验证成功之后，DB2 UDB 必须判断用户是否具有执行操作所需的权限。这个过程称为“授权”，如图 14.2 上半部分所示。

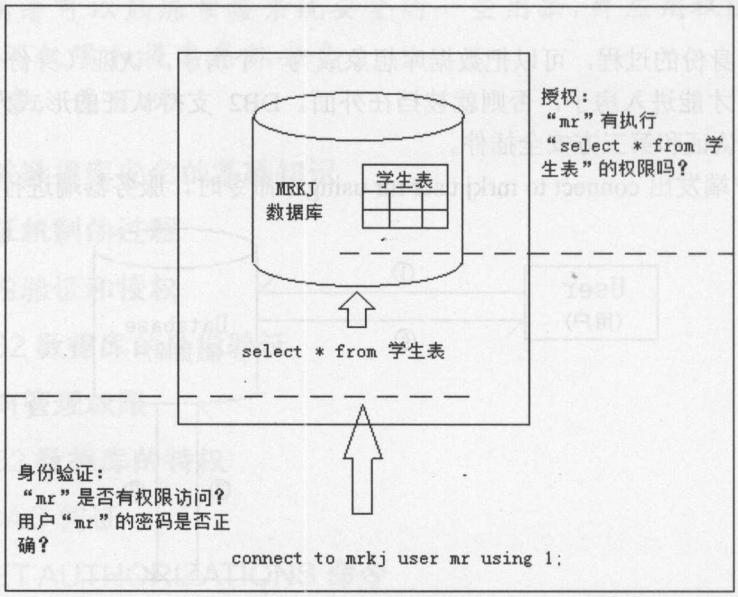


图 14.2 身份验证和授权

只有当用户具有执行指定函数的权限时，它们才能成功执行 SQL 或 XQuery 语句。要创建表，必须授权用户创建表的权利；要改变表，必须授予用户改变表的权利。

DB2 UDB 使用 3 种内部机制来判断授权是否成功——管理权限、特权和 LBAC 凭证。

14.4　DB2 身份验证

为了确保数据库的安全性，用户必须提供用户名和密码，然后才能连接到数据库。

14.4.1　DB2 身份验证方式

DB2 使用身份验证方式决定在什么地方进行身份验证。例如，在客户机-服务器环境中，是在客户机还是服务器检验用户的 ID 和密码？在客户机-网关-主机环境中，是在客户机还是主机检验用户的 ID 和密码？

表 14.1 总结了可用的 DB2 身份验证方式。在客户机-服务器环境中，身份验证选项在客户机上设置，而不是在服务器上设置。在客户机-网关-主机环境中，身份验证选项在客户机和网关上设置，而不是在主机上设置。

表 14.1　DB2 身份验证方式

类　型	描　述
服务器（SERVER）	身份验证在服务器上运行
服务器加密（SERVER_ENCRYPT）	身份验证在服务器上运行，密码在客户机上进行加密，然后再发送到服务器
客户机（CLIENT）	身份验证在客户机上进行
KERBEROS	由 Kerberos 安全软件执行身份验证
KERBEROS 加密（KEB_SERVER_ENCRYPT）	如果客户机设置是 KERBEROS，那么由 Kerberos 安全软件执行身份验证：否则使用 SERVER_ENCRYPT
数据加密（DATA_ENCRYPT）	身份验证在服务器上进行，服务器接受加密的用户 ID 和密码，并对数据进行加密；这个选项的操作方式与 SERVER_ENCRYPT 相同，但是数据也要加密
数据加密一致（DATA_ENCRYPT_CMP）	身份验证方式与 DATA_ENCRYPT 相同，但是允许不支持 DATA_ENCRYPT 的老式客户机使用 SERVER_ENCRYPT 身份验证进行连接；在这种情况下，数据不进行加密；如果进行连接的客户机支持 DATA_ENCRYPT，就会进行数据加密，而不能降级到 SERVER_ENCRYPT 身份验证；这个身份验证方式只在服务器的数据库管理程序配置文件中是有效的，而且在客户机或网关实例上使用 CATALOG DATABASE 时是无效的
GSS 插件（GSSPLUGIN）	身份验证方式由一个外部 GSS-API 插件决定
GSS 服务器加密（GSS_SERVER_ENCRYPT）	身份验证方式由一个外部 GSS-API 插件决定；在客户机不支持服务器的 GSS-API 插件之一的情况下，使用 SERVER_ENCRYPT 身份验证

DB2 的安全插件架构：

DB2 V8.2 引入了安全插件的概念，这个概念在 DB2 V8.2 之后的版本中得到了进一步增强。通过使用标准的 GSS-API 调用，用户可以编写一个安全插件，并将对用户 ID 进行身份验证的工作交给一个外部安全程序。这种方式的一个示例是 DB2 本身的 KERBEROS 身份验证。在一台机器上安装 DB2

ESE 或应用程序开发客户机时，安装过程中将有一些示例应用程序代码放入实例目录，如果查看下面的路径：C:\Program Files\IBM\SQLLIB\samples\security\plugins，就会看到如图 14.3 所示的安全插件示例。

名称	修改日期	类型	大小
AuthidMapper.c	2009/5/30 20:14	VisualStudio.c.11...	9 KB
bldplugin.bat	2009/5/30 20:14	Windows 批处理...	2 KB
combined.c	2009/5/30 20:14	VisualStudio.c.11...	40 KB
group_file.c	2009/5/30 20:14	VisualStudio.c.11...	19 KB
gssapi_simple.c	2009/5/30 20:14	VisualStudio.c.11...	49 KB
IBMkrb5.c	2009/5/30 20:14	VisualStudio.c.11...	76 KB
IBMLDAPauthclient.c	2009/5/30 20:14	VisualStudio.c.11...	12 KB
IBMLDAPauthserver.c	2009/5/30 20:14	VisualStudio.c.11...	14 KB
IBMLDAPconfig.c	2009/5/30 20:14	VisualStudio.c.11...	17 KB
IBMLDAPgroups.c	2009/5/30 20:14	VisualStudio.c.11...	37 KB
IBMLDAPutils.c	2009/5/30 20:14	VisualStudio.c.11...	30 KB
IBMLDAPutils.h	2009/5/30 20:14	VisualStudio.h.1...	9 KB
makefile	2009/5/30 20:14	文件	3 KB
README	2009/5/30 20:14	文件	4 KB

图 14.3　安全插件

14.4.2　其他身份验证参数

查看配置文件中的身份验证参数的命令为：

```
GET DBM CFG;
```

执行该命令后将看到如图 14.4 所示的验证参数（由于篇幅关系只截取了部分参数）。

```
---------- 输入的命令 ----------
GET DBM CFG;
------------------------------------------------------------------------
GET DBM CFG

   数据库管理器配置

 节点类型                          = 带有本地客户机和远程客户机的企业服务器版

数据库管理器配置发行版级别                = 0x0d00

最大打开文件数                (MAXTOTFILOP) = 16000
CPU 速度（毫秒/指令）             (CPUSPEED) = 1.535119e-007
通信带宽（MB/秒）              (COMM_BANDWIDTH) = 1.000000e+002
```

图 14.4　验证参数

如果查看 DB2 V9.7 示例中的 DBM CFG，会看到影响 DB2 对用户 ID 进行身份验证操作的各种设置，有针对标准操作系统用户 ID 身份验证的设置（CLIENT、SERVER、SERVER_ENCRYPT、DATA_ENCRYPT、DATA_ENCRYPT_CMP），还有将身份验证工作交给外部程序的插件（KERBEROS、KRB_SERVER_ENCRYPT、GSSPLUGIN、GSS_SERVER_ENCRYPT）。

（1）DB2 V9.1 以上的版本能够根据用户是试图连接数据库，还是执行实例和实例级操作，来指

定不同的身份验证机制。在默认情况下，实例对于所有实例级和连接级请求使用一种身份验证方式。这由数据库管理程序配置参数 AUTHENTICATION 来指定。DB2 V9.7 中引入了数据库管理程序配置参数 SRVCON_AUTH，这个参数专门处理对数据库的连接。

例如，如果在 DBM 配置中将身份验证参数 AUTHENTICATION 设置为“SERVER_ENCRYPT”，而将 SERVCON_AUTH 设置为“KERBEROS”，那么在连接实例时会使用 SERVER_ENCRYPT 进行验证，但是在连接数据库时会使用 KERBEROS 身份验证。如果在服务器上没有正确初始化 KERBEROS，但是提供了有效的用户 ID/密码，那么允许这个用户连接实例，但是不允许他连接数据库。

（2）CLNT_PW_PLUGIN 这个参数在客户端的 DBM CFG 中指定。它指定用来进行客户机和本地身份验证的客户机插件的名称。

（3）GROUP_PLUGIN 默认值是空（NULL）。将它设置为用户定义的插件就会调用这个插件进行所有组枚举，而不依赖于操作系统的组查找。这与后面讨论的授权部分相关。

（4）LOCAL_GSSPLUGIN 这个参数指定默认的 GSS-API 插件库的名称，当数据库管理程序配置的身份验证参数值设置为 GSSPLUGIN 或 GSS_SERVER_ENCRYPT 时，这个库用来进行实例级的本地授权。

（5）SRV_PLUGIN_MODE(YES/NO)这个参数的默认设置是 NO。当改为 YES 时，会以 FENCED 模式启动 GSS-API 插件，其工作方式类似于 FENCED 存储过程。FENCED 插件的崩溃不会导致 DB2 实例崩溃。在插件还处于开发阶段时，建议以 FENCED 模式运行它们，这样的话，插件中的逻辑问题和内存泄露就不会导致实例崩溃。在确定插件是可靠的之后，应该以 UNFENCED 模式运行，以提高其性能。

（6）SRVCON_GSSPLUGIN_LIST 是一个插件列表，当使用 KERBEROS、KRB_SERVER_ENCRYPT、GSSPLUGIN 或 GSS_SERVER_ENCRYPT 时，服务器上的数据库管理程序在身份验证期间将使用这些插件。列表中的每个插件应该用逗号（,）分隔，它们之间没有空格。插件按照优先次序列出，首先使用列表中的第一个插件对发送的用户 ID/密码进行身份验证。只有当列出的所有插件都返回了错误时，DB2 才会向用户返回身份验证错误。

（7）SERVCON_PW_PLUGIN 在指定 CLIENT、SERVER 或 SERVER_ENCRYPT 身份验证时，这个参数允许用户修改 DB2 用来检验用户 ID 和密码的默认身份验证方法。在默认情况下，它的值是 NULL，因此使用默认的 DB2 方法。

（8）CATALOG_NOAUTH （YES/NO）默认值是 NO。将这个参数修改为 YES，就允许不属于 SYSADM、SYSCTRL 或 SYSMAINT 组成员的用户修改机器上的 DATABASE、NODE、ADMIN 和 DCS 编目。如果登录的用户使用不可信客户机，或者登录所用的用户 ID 不允许连接数据库或实例，但是用户又必须在客户机上进行编目，只有在这种情况下这个参数才是有用的。

（9）如果将 FED_NOAUTH 设置为 YES，身份验证设置为 SERVER 或 SERVER_ENCRYPT，联邦设置为 YES，那么在实例上避免进行身份验证。它假设身份验证在数据源上进行。当 FED_NOAUTH 设置为 YES 时，系统会发出警告。在客户机和 DB2 服务器上都不进行身份验证。直到 SYSADM 身份验证名称的任何用户都拥有联邦服务器的 SYSADM 权限。

14.4.3　在客户机-服务器环境中设置身份验证

下面以客户机-服务器环境为例，说明如何设置身份验证。

【例 14.1】　如何将网络中的 DB2 服务器数据库添加到 DB2 客户机。

1. 测试环境

局域网中的主机 MRJ（这里将它看成是服务器）上安装有 DB2 服务器程序，有一个实例 DB2，在实例 DB2 中有一个数据库 MRK（数据库已启动）；局域网中的另一台主机 MR-PC（这里将它看成是客户机）安装有 DB2 服务器程序（或者 DB2 客户机程序）。

注意

如果要将 DB2 服务器端数据库添加到 DB2 客户端，服务器端必须安装 DB2 服务器程序，客户端必须已安装 DB2 客户端程序或 DB2 服务器程序。

2. 实验内容

通过身份验证，将服务器 MRJ 上的数据库 MRK 添加到客户机 MR-PC 上。也就是说，在客户机 MR-PC 上建立数据库 MRK 的镜像，并不在客户机 MR-PC 上真正存储数据库 MRK。

在客户机上可以将被添加的数据库像本地数据库一样使用（例如，将该数据库添加为数据源或将该数据库连接到.NET 或 J2EE 等应用程序）。

3. 在数据库服务器上设置身份验证方式（如果选择默认的身份验证则不需要这一步）

实现步骤如下。

（1）在数据库服务器环境下（这里是 MRJ），在“控制中心”窗口选择“所有系统”→“系统名”（这里是“MRJ”）→“实例”→“实例名”（这里是“DB2”）文件夹，在“实例名”（这里是 DB2）文件夹单击鼠标右键，在弹出的快捷菜单中选择“配置参数”命令，从而打开“DBM 配置”对话框，如图 14.5 所示。

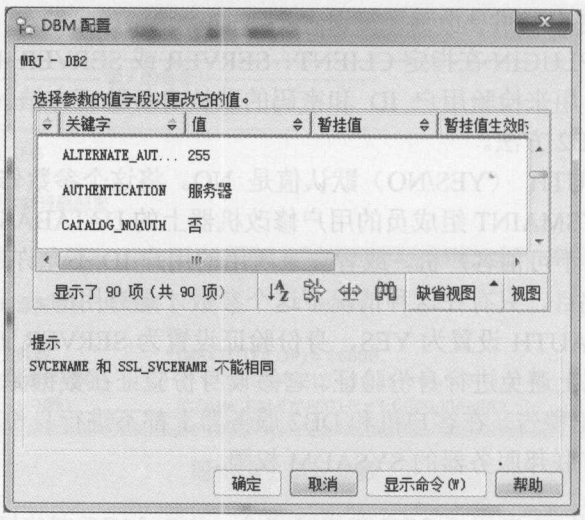


图 14.5　配置身份验证方式

（2）在“DBM 配置”对话框中单击“管理”目录下的 AUTHENTICATION 文本框后面的“...”按钮，如图 14.6 所示，在弹出的“更改 DBM 配置参数-AUTHENTICATION”对话框中，在“认证”下面的单选按钮中选择某种身份验证方式（例如，“服务器加密”），然后单击“确定”按钮，如图 14.7

所示。

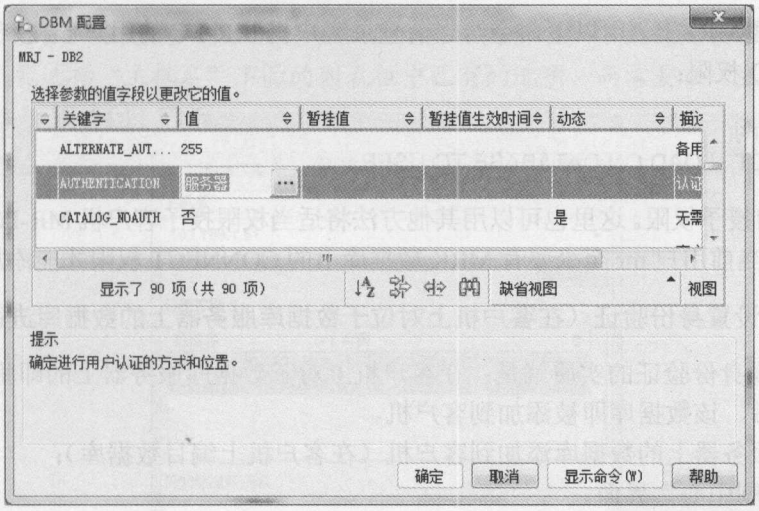


图 14.6　配置身份验证方式

这时如图 14.8 所示的对话框中的参数 AUTHENTICATION 所在的行中就出现了身份验证方式的暂挂值（例如，“服务器加密”）。在实例重新启动后修改的参数才能生效。

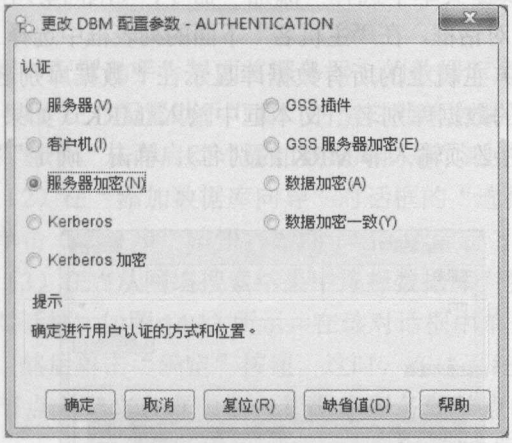


图 14.7　“更改 DBM 配置参数-AUTHENTICATION”对话框

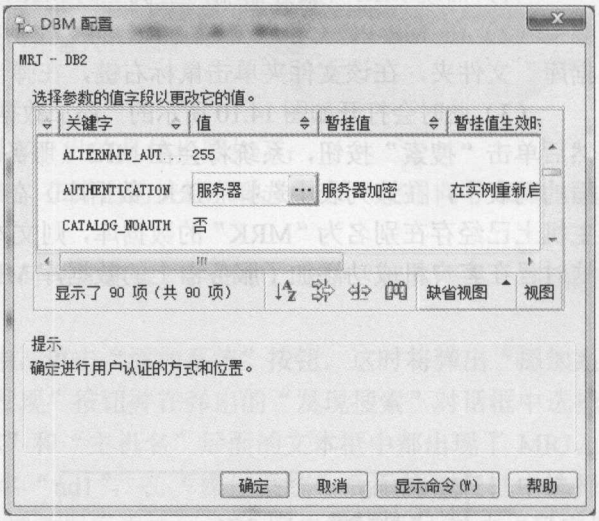


图 14.8　修改暂挂值

步骤（1）和（2）的操作过程也可以在数据库服务器上用以下命令完成：

```
UPDATE DBM CFG USING AUTHENTICATION SERVER_ENCRYPT;
```

（3）在“命令编辑器”中执行下面的命令，使参数 ATUTHENTICATION 的修改生效：

```
DB2STOP;
DB2START;
```

这时，服务器 MRJ 的实例 DB2 的身份验证方式就改为“服务器加密”。

4．在数据库服务器上把即将添加的数据库上的适当权限授予客户机

在数据库服务器 MRJ 上，用以下命令给当前登录客户机 MR-PC 的用户 mr 授予 MRK 数据库上的 CONNECT 和 LOAD 权限：

```
CONNECT TO mrkj;
GRANT CONNECT , LOAD ON DATABASE TO USER mr;.
```

后面会讲到如何授予权限。这里也可以用其他方法将适当权限授予客户机 MR-PC 上的当前用户 mr。客户机 MR-PC 上的当前用户 mr 至少要有 MRK 数据库上的 CONNECT 权限才能添加和连接数据库。

5．在客户机上设置身份验证（在客户机上对位于数据库服务器上的数据库进行编目）

在客户机上设置身份验证的实质就是：在客户机上对于数据库服务器上的即将被添加的数据库编目，编目该数据库后，该数据库即被添加到客户机。

有 3 种方法将服务器上的数据库添加到客户机（在客户机上编目数据库）：

（1）使用“控制中心”界面。

（2）使用“配置助手”界面。

（3）使用 SQL 命令。

用“控制中心”界面将服务器上的数据添加到客户机的实现步骤如下：

（1）启动 MRJ（服务器端）主机上的 MRK 数据库。

（2）在 MR-PC（客户端）主机上的“控制中心”对象视图中，选择“控制中心”下的“所有数据库”文件夹，在该文件夹单击鼠标右键，在弹出的快捷菜单中选择“添加”命令，如图 14.9 所示。

（3）此时会打开如图 14.10 所示的“添加数据库”对话框，在“主机名”下面的列表框中选择 MRJ，然后单击“搜索”按钮，系统将会在 MRJ（服务器端）主机上的所有数据库显示在“数据库别名”下面的列表中，在该列表中选择 MRK 数据库。在“新的数据库别名”文本框中输入 MRK（如果 MRJ 主机上已经存在别名为“MRK”的数据库，则文本框中必须输入非 MRK 的别名）。单击“确定”按钮，这时已在客户机成功添加了服务器上的数据库 MRK。

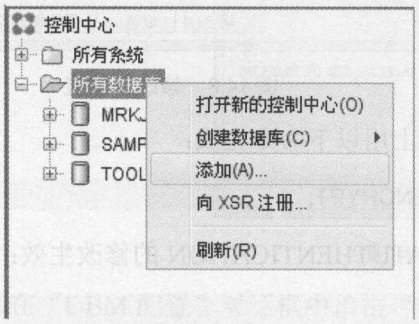


图 14.9　打开快捷菜单

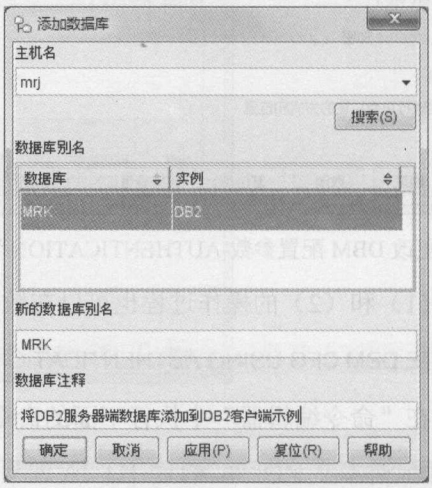


图 14.10　“控制中心”的“添加数据库”对话框

说明

本例中服务器 MRJ 在局域网中，所以它会显示在“主机名”下面的列表框中。如果服务器不在局域网中，则不能选择“主机名”下面的列表框中已有的选项，而需要在“主机名”下面的列表框中输入 IP 地址，如图 14.11 所示。

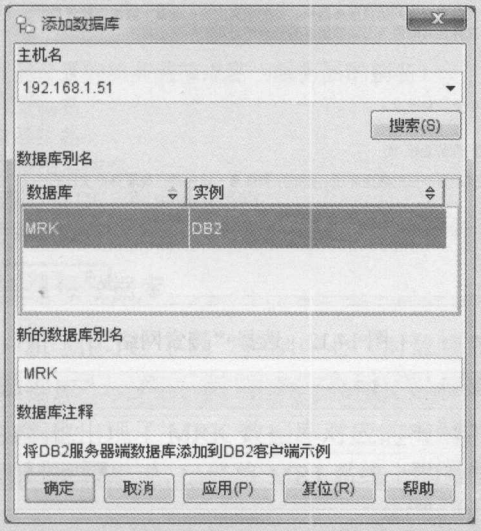


图 14.11　在“主机名”中输入 IP 地址

用“配置助手”界面将服务器上的数据库添加到客户机的具体步骤如下：

（1）在“配置助手”窗口的空白处单击鼠标右键，在弹出的快捷菜单中选择“使用向导来添加数据库”命令，从而启动“添加数据库向导”对话框。

（2）在“添加数据库向导”对话框的“选择要如何设置连接”界面中，选中“搜索网络”单选按钮，单击“下一步”按钮，如图 14.12 所示。

（3）在“从网络搜索结果中选择数据库”界面中，单击“添加系统”按钮，这时将弹出“添加系统”对话框，如图 14.13 所示。在该对话框中单击“发现”按钮并在弹出的“发现搜索”对话框中选择 MRJ，然后单击“确定”按钮，这时，在“系统名称”和“主机名”后面的文本框中都出现了 MRJ。在“节点名”文本框中输入数据库服务器节点的别名“nd1”，在“操作系统”后面的列表框中选择 Windows，在“注释”文本框中输入“将 DB2 服务器端数据库添加到 DB2 客户端示例”，单击“确定”按钮，这时在图 14.14 所示的对话框中就出现了局域网中数据库的目录树，在该目录树中选择“已知系统”→“系统名”（这里是 MRJ）→“实例名”（这里是 DB2）→“本地数据库”→MRK 数据库，单击“完成”按钮。

说明

如果数据库服务器不存在于局域网中（远程数据库），则在如图 14.13 所示的“添加系统”对话框中不能单击“发现”按钮帮助填写服务器信息，而需要在“系统名称”对话框中手动输入服务器的 IP 地址，并输入服务器的相关信息。

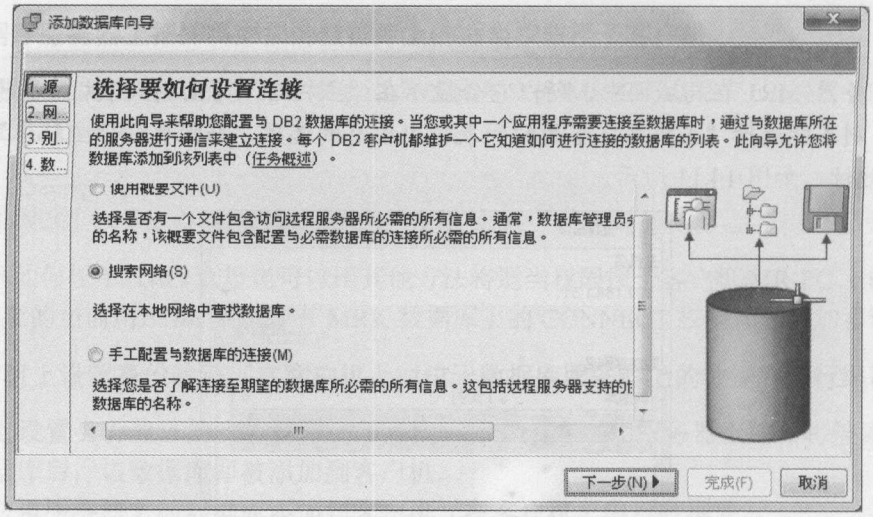


图 14.12　选择“搜索网络”

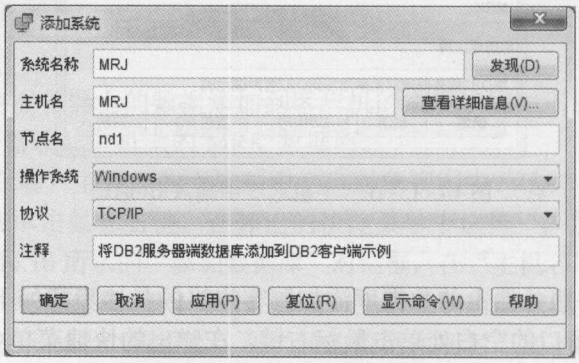


图 14.13　“添加系统”对话框

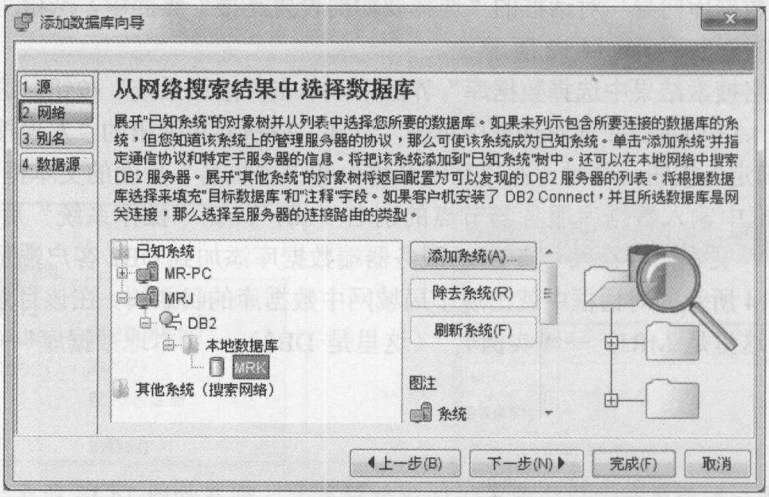


图 14.14　在局域网中的数据库列表中选择要添加的数据库

（4）在弹出的“添加数据库确认”对话框中单击“关闭”按钮，这时已在客户机成功添加了服务

器上的数据库 MRK。

用 SQL 命令将数据库服务器上的数据库添加到客户机上的方法如下：

在客户机节点上使用 CATALOG DATABASE 命令来对位于数据库服务器上的数据库进行编目，编目后该数据库就添加到了客户机。

在客户机上发出以下命令将对新添加的名称为 MRK 的服务器数据库进行编目：

```
CATALOG ADMIN TCPIP
NODE nd1          --在客户机上设置的服务器节点名（服务器的别名）
REMOTE MRJ        --服务器的主机名
SYSTEM MRJ        --服务器的系统名
OSTYPE WIN        --服务器的操作系统名
WITH '将 DB2 服务器端数据库添加到 DB2 客户端示例';
                  --新添加的数据库注释
```

6. 查看新添加的数据库并连接该数据库

（1）添加数据库后，在客户机 MR-PC 上的“控制中心”对象视图中刷新“所有系统”和“所有数据库”文件夹；这时，“所有数据库”文件夹下就出现了 MRK 数据库；在“所有系统”文件夹下也出现了系统名 MRJ，在 MRJ 目录下也出现了 MRK 数据库对应的实例名（这里是“TCP2A419（DB2）”）和数据库名 MRK，如图 14.15 所示。也就是说，已经将 MRJ 系统上的数据库添加到了本地系统 MR-PC。

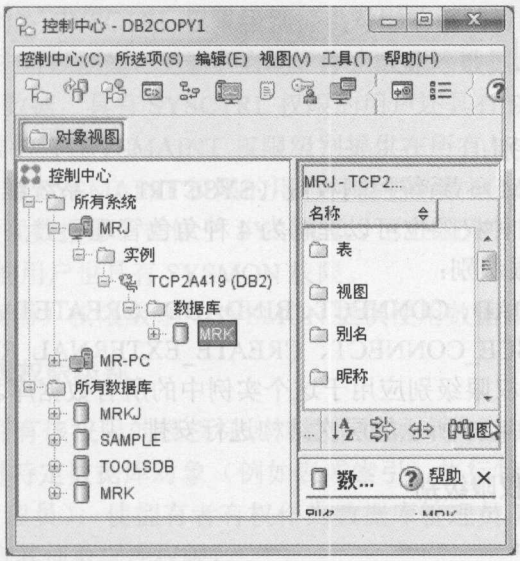


图 14.15　在客户端添加了服务器端数据库

说明

系统自动产生的实例名“TCP2A419（DB2）”中“DB2”是数据库服务器 MRJ 上的 MRK 数据库所在的实例的实例名，“TCP2A419”是 MRJ 上 DB2 实例节点名（实例别名），这个别名由系统自动产生。如果除去已添加的 MRK 数据库后再次添加该数据库，可能自动产生不同于上一次添加 MRK 数据库时的实例节点名。实例节点名可以更改。

（2）在“控制中心”对象视图中选择“所有系统”→“系统名”（这里是MRJ）→“实例”→“实例名”（这里是“TCP2A419（DB2）”）→“数据库”→MRK 文件夹，在该文件夹上单击鼠标右键，在弹出的快捷菜单中选择“连接”命令，或在“控制中心”对象视图中的“所有数据库”→MRK 上单击鼠标右键，在弹出的快捷菜单中选择“连接”命令。



图 14.16　连接新添加的数据库

（3）在弹出的“连接”对话框中输入用户标识和密码，如图 14.16 所示，然后单击“确定”按钮。如果用户标识和密码正确，这时就从客户机连接到了服务器上的数据库 MRK。

14.5　管 理 权 限

拥有了管理权限的人可以管理控制数据库管理器的任务并负责数据的安全性和完整性。DB2 UDB 定义了一个管理权限级别层次的结构，用于将一组预先确定的管理权限授予用户账号或组。这些管理权限包括能够对数据库进行备份、迫使用户离开系统、查看表数据等。

14.5.1　管理权限级别

（1）有 4 个实例权限级别：

系统管理权限（SYSADM）、系统控制权限（SYSCTRL）、系统维护权限（SYSMAINT）、系统监控权限（SYSMON）。这 4 种权限也可以理解为 4 种角色管理。

（2）有 10 个数据库权限级别：

DBAMD、SECADM、LOAD、CONNECT、BINDADD、CREATETAB、CREATE_NOT_FENCED、IMPLICIT SCHEMA、QUIESCE_CONNECT、CREATE_EXTERNAL_ROUTINE。

（3）在实例级上定义的权限级别应用于这个实例中的所有数据库。

管理权限级别按照如图 14.17 所示的层次结构进行安排。

1. 在实例级操作的管理权限级别

SYSADM（系统管理员）：

- ☑ 在这个层次结构的顶部是 SYSADM 权限级别，这是 DB2 UDB 中最高的权限级别。具有管理权限级别 SYSADM 和 DBADM 的人隐式具有对所有对象的特权，并控制有权访问数据库管理器的人以及其访问权的范围。
- ☑ SYSADM 权限级别提供对数据库管理器所创建和维护的全部资源的控制。系统管理员拥有所有 DBADM、SYSCTRL、SYSMAINT 和 SYSMON 权限，并有权授予和撤销 DBADM 权限和 SECADM 权限。拥有 SYSADM 权限的用户负责控制数据库管理器，并确保数据的安全和完整性。SYSADM 权限在数据库内提供隐式 DBADM 权限，但不在数据库内提供隐式 SECADM

权限。

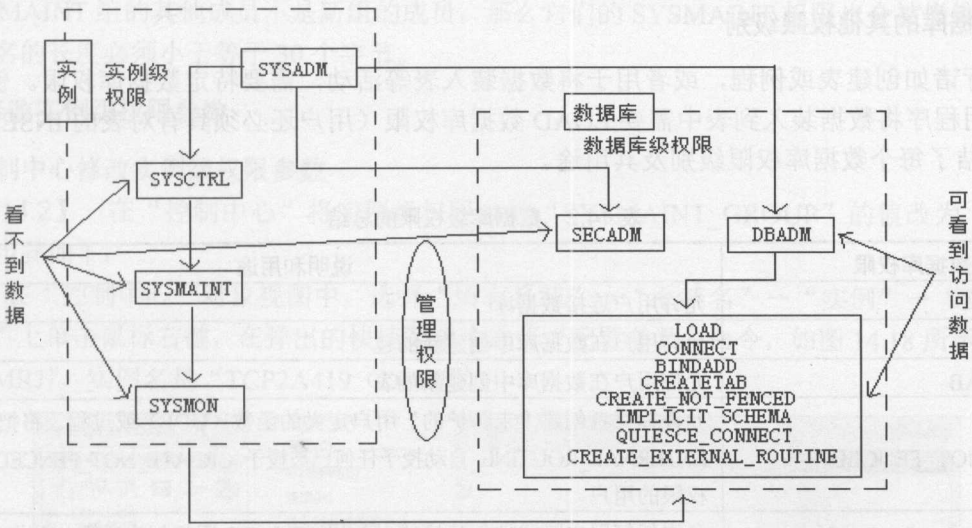


图 14.17　DB2 UDB 中的权限级别层次结构

2．在实例级操作的系统控制权限级别

☑ SYSCTRL（系统控制）：SYSCTRL 权限级别提供对影响系统资源的操作的控制。例如，具有 SYSCTRL 权限的用户可以创建、更新、启动、停止或删除数据库。此用户还可以启动或停止实例，但不能访问表数据。具有 SYSCTRL 权限的用户还具有 SYSMON 权限。

☑ SYSMAINT（系统维护）：SYSMAINT 权限级别提供在所有与实例关联的数据库上执行维护操作所需的权限。具有 SYSMAINT 权限的用户可以更新数据库配置、备份数据库或表空间、复原现有数据库并检视数据库。类似于 SYSCTRL，SYSMAINT 不提供对表数据的访问。具有 SYSMAINT 权限的用户也具有 SYSMON 权限。

☑ SYSMON（系统监视器）权限级别：SYSMON 提供使用数据库系统监视器所需的权限。

3．在数据库级操作的管理权限级别

每个数据库权限都允许拥有该权限的授权标识对整个数据库执行某种特定类型的操作。数据库权限与特权不同，后者允许对特定数据库对象（例如表或索引）执行特定操作。

☑ DBADM（数据库管理员）：使拥有者有权作为数据库管理员工作。特别是它授予拥有者除 SECADM 以外的所有其他数据库权限。

☑ SECADM（安全管理员）：使拥有者能够充当安全管理员，并且能够创建和删除安全对象、授予和撤销安全对象的权限或特权，并转移对象的所有权。安全管理员管理可信上下文、审计策略、数据库角色和 LBAC 保护的数据。并且是授予和撤销 SETESESSIONUSER 特权所需的权限。具有 SECADM 权限的用户可以转移不属于他们对象的所有权。他们可以使用 AUDIT 语句将审计策略与服务器中的特定数据库或数据库对象关联。

☑ SECADM 权限没有访问存储在表中的数据库的固有特权，也没有其他附加固有特权。SECADM 权限只能由具有 SYSADM 权限的用户授予。可以将 SECADM 权限授予用户，但不能授予组、

角色或 PUBLIC。

4. 数据库的其他权限级别

要执行诸如创建表或例程，或者用于将数据装入表等活动，需要特定数据库权限。例如，使用 LOAD 实用程序将数据装入到表中需要 LOAD 数据库权限（用户还必须具有对表的 INSERT 特权）。表 14.2 总结了每个数据库权限级别及其用途。

表 14.2 数据库级权限的总结

数据库权限	说明和用途
CONNECT	允许用户连接数据库
BINDADD	允许用户在数据库中创建新的包
CREATETAB	允许用户在数据库中创建新的表
CREATE_NOT_FENCED	允许拥有者创建“未防护的”用户定义的函数（UDF）或过程。将把 CREATE_EXTERNAL_ROUTINE 自动授予任何已被授予 CREATE_NOT_FENCED_ROUTINE 权限的用户
IMPLICIT SCHEMA	允许任何用户隐式地创建模式（使用 CREATE 语句创建对象，并指定尚不存在的模式名）。SYSIBM 成为隐式创建的模式的所有者，并且授予 PUBLIC 在此模式中创建对象的特权
LOAD	允许拥有者将数据装入到表中
QUIESCE_CONNECT	允许拥有者在数据库处于停顿状态时访问该数据库
CREATE_EXTERNAL_ROUTINE	允许用户注册外部例程（用 C 和 Java 等外部语言编写的例程）

只有拥有 SYSADM 权限的授权标识才能授予 SECADM 和 DBADM 权限。所有其他权限都可以由拥有 SYSADM 或 DBADM 权限的授权标识授予。

创建数据库时，对于新数据库，将自动地把下列数据库权限授予 PUBLIC：CREATETAB、BIDADD、CONNECT、IMPLICIT_SCHEMA。

说明

DB2 UDB 将关于数据库级权限的信息存在系统编目视图 SYSCAT.DBAUTH 中。

在向用户或组授予权限和特权时必须小心，因为 DB2 UDB 允许将这些权限和特权授予不存在的账号。如果以后创建了同名账号，那么这个账号会自动获得以前授予的所有权限和特权。

14.5.2 授予或撤销实例级权限

1. 建立实例级权限的办法

将在外部安全设施中定义的用户组赋给相关的实例级权限参数（SYSADM_GROUP、SYSCTRL_GROUP、SYSMAINT_GROUP、SYSMON_GROUP）。例如，如果希望用户账号 MR1 具有 SYSMAINT 权限，那么可以将 MR1 放进 MAINT 组，然后将实例参数 SYSMAINT_GROUP 更新为 MAINT。这样，MAINT 组中的任何用户都将具有 SYSMAINT 权限。要从 MR1 那里撤销 SYSMAINT 特权，只需从

MAINT 组中删除它，或者将 SYSMAINT_GROUP 参数的值改为另一个不包含它的组。在后一种情况下，如果 MAINT 组的其他成员不是新组的成员，那么它们的 SYSMAINT 权限也会被撤销。在所有平台上，组名的长度必须小于等于 30 个字节。

2．修改实例级权限参数

在控制中心修改实例级权限参数

【例 14.2】　在“控制中心”将实例级权限参数“SYSMAINT_GROUP”的值改为“MAINT”。实现步骤如下：

（1）在“控制中心”对象视图中，选择“所有系统”→“系统名”→“实例”→“实例名”，在“实例名”上单击鼠标右键，在弹出的快捷菜单中选择“配置参数”命令，如图 14.18 所示（图中的系统名是“MRJ”，实例名是“TCP2A419（DB2）”）。

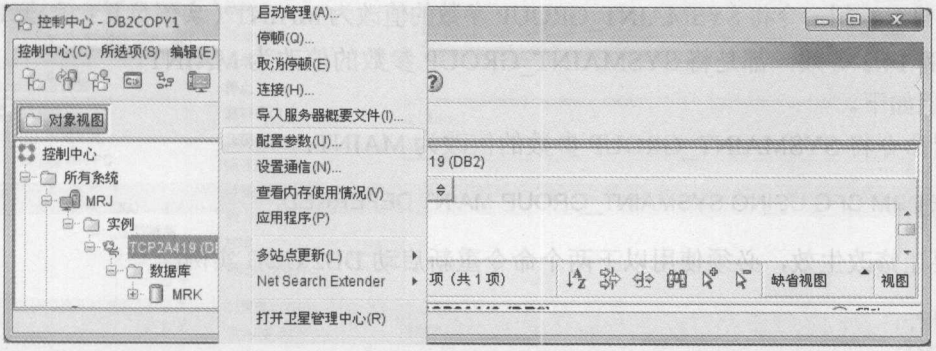


图 14.18　“控制中心”的“配置参数”菜单

（2）在弹出的“DBM 配置”对话框中找到“管理”目录下的“SYSMAINT_GROUP”的权限级别参数。单击该参数文本框后面的“…”按钮，在弹出的对话框中，将 SYSMAINT_GROUP 参数的值改为 MAINT。然后在所有的对话框中都单击“确定”按钮，如图 14.19 和图 14.20 所示。

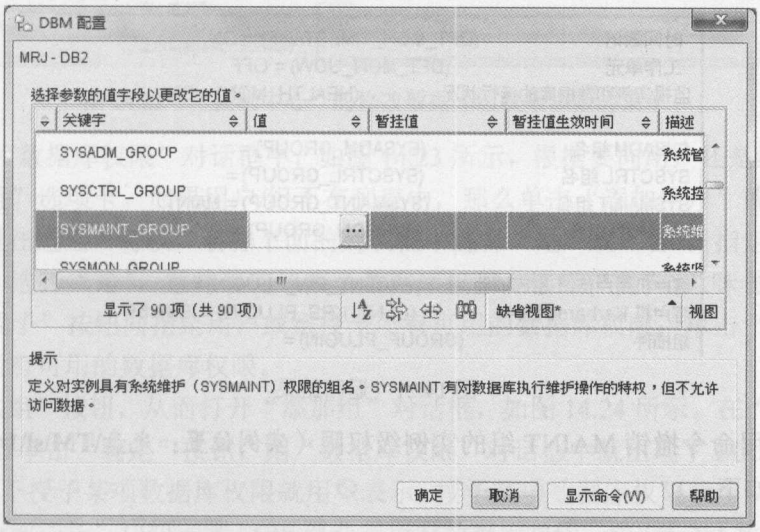


图 14.19　修改 SYSMAINT_GROUP 参数

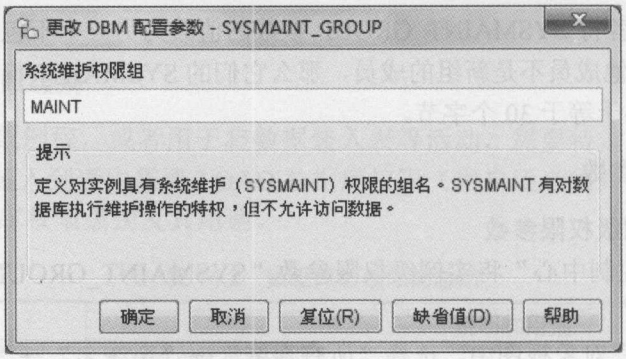


图 14.20　修改 SYSMAINT_GROUP 参数

用命令修改实例级权限参数

【例 14.3】　用命令将 SYSMAINT_GROUP 参数的值改为 MAINT（**实例位置：光盘\TM\sl\14\1**）。

本例与例 14.2 等效，都是将 SYSMAINT_GROUP 参数的值改为 MAINT。

实现步骤如下。

（1）用命令将 SYSMAINT_GROUP 参数的值改为 MAINT：

```
UPDATE DBM CFG USING SYSMAINT_GROUP MAINT DEFERRED;
```

（2）要让修改生效，必须使用以下两个命令重新启动 DB2 UDB 实例：

```
DB2STOP;
DB2START;
```

（3）要确认修改已经生效，可以通过执行以下命令来查看参数的值：

```
GET DBM CFG;
```

返回结果如图 14.21 所示。

```
 时间戳记                    (DFT_MON_TIMESTAMP) = ON
 工作单元                    (DFT_MON_UOW) = OFF
监视实例和数据库的运行状况          (HEALTH_MON) = ON

SYSADM 组名                 (SYSADM_GROUP) =
SYSCTRL 组名                (SYSCTRL_GROUP) =
SYSMAINT 组名               (SYSMAINT_GROUP) = MAINT
SYSMON 组名                 (SYSMON_GROUP) =

客户机用户标识-密码插件          (CLNT_PW_PLUGIN) =
客户机 Kerberos 插件        (CLNT_KRB_PLUGIN) = IBMkrb5
组插件                      (GROUP_PLUGIN) =
```

图 14.21　返回结果

【例 14.4】　用命令撤销 MAINT 组的实例级权限（**实例位置：光盘\TM\sl\14\2**）。

具体命令语句如下：

```
UPDATE DBM CFG USING SYSMAINT_GROUP NULL DEFERRED;
```

14.5.3　授予或撤销数据库级权限

1. 在控制中心授予或撤销数据库级权限

【例 14.5】　在“控制中心”修改 MRKJ 数据库的数据库级权限参数。

实现步骤如下：

（1）在“控制中心”对象视图中，选择“所有系统”→“系统名”→“实例”→“实例名”→“数据库”→“数据库名”，单击鼠标右键，在弹出的快捷菜单中选择“权限”命令，如图 14.22 所示（图中系统名是“MR-PC”，实例名是“DB2”，数据库名是“MRKJ”）。

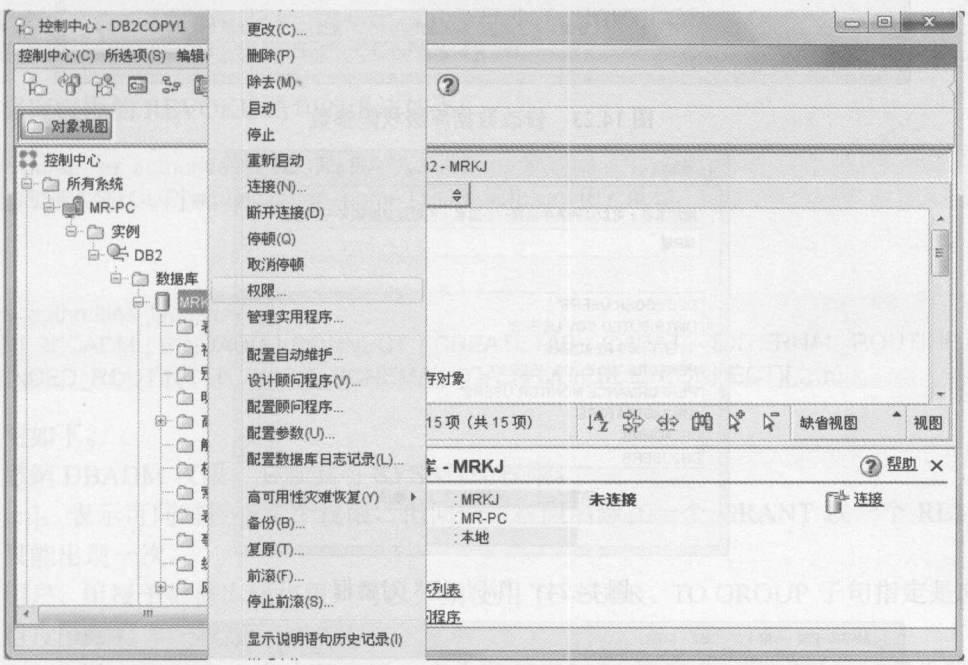


图 14.22　“控制中心”的修改数据库级权限参数的菜单

（2）在弹出的“数据库权限”对话框中，如图 14.23 所示，根据要向用户还是组授予数据库权限，选择“用户”或“组”选项卡。如果用户/组不在列表中，那么单击“添加用户”或“添加组”按钮，添加用户或组。通过在适当“特权”名称下面的列表框中选择“是”或“否”，指定应该向用户或组授予哪些数据库权限。选择“是”，意味着应该授予数据库权限，选择“否”，意味着应该不授予数据库权限；单击“全部授予”按钮向指定用户或组授予所有可用的数据库权限；单击“全部撤销”按钮从指定用户或组撤销所有可用的数据库权限。

这里单击“添加组”按钮，从而打开“添加组”对话框，如图 14.24 所示。在“添加组”对话框中输入组名 MRS，然后单击“确定”按钮，则“数据库权限”对话框中就出现了 MRS 组，且 MRS 组没有任何数据库权限（不授予某项数据库权限就用⊘表示，授予某项数据库权限就用✓表示），如图 14.25 所示。这里单击“全部授予”按钮，则 MRS 组获得所有数据库权限（SECADM 数据库权限除外）。

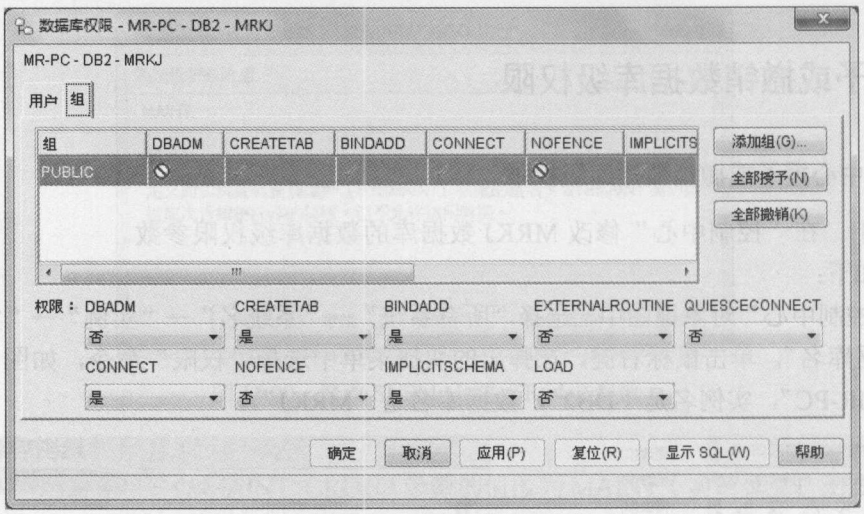


图 14.23　修改数据库级权限参数

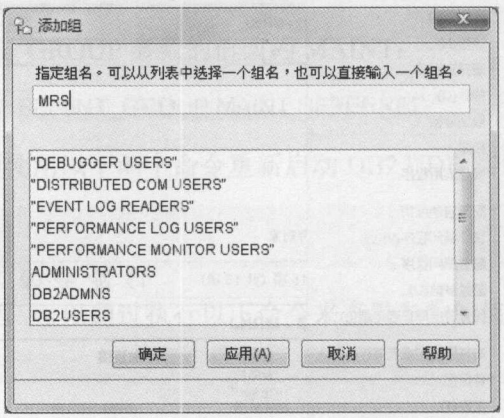


图 14.24　“添加组”对话框

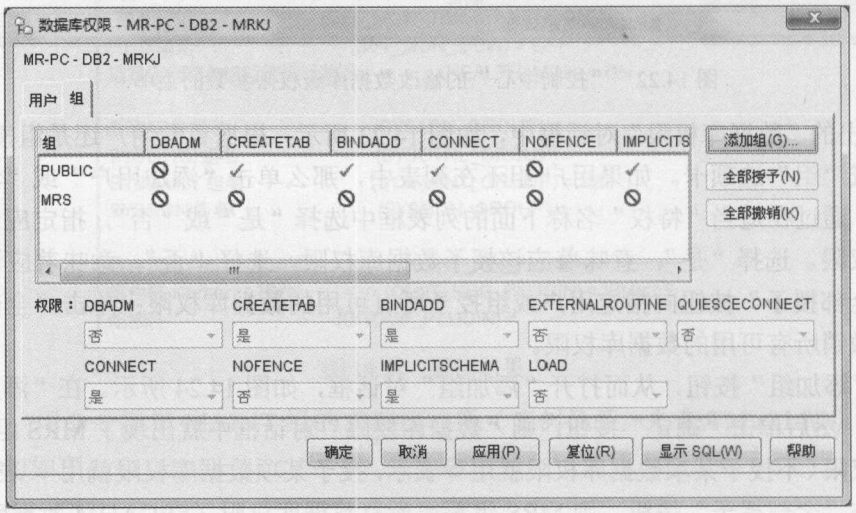


图 14.25　添加后的结果

2. 用命令授予或撤销数据库级权限

也可使用 GRANT 语句将数据库级权限（如 DBADM、LOAD 和 CREATETAB）授予用户或组。使用 REVOKE 语句撤销数据库级权限。

授予数据库权限的 GRANT 语句的语法如下：

```
GRANT database_authorities_type ON DATABASE
TO {[USER | GROUP] authorization_name | PUBLIC} [,...n]
```

其中：

```
database_authorities_type ::=
    {DBADM | SECADM | BINDADD | CONNECT | CREATETAB|
CREATE_EXTERNAL_ROUTINE | CREATE_NOT_FENCED_ROUTINE |
IMPLICIT_SCHEMA | LOAD | QUIESCE_CONNECT}[,..n]
```

撤销数据库权限的 REVOKE 语句的语法如下：

```
REVOKE database_authorities_type ON DATABASE
FROM {[USER | GROUP] authorization_name | PUBLIC}[,...n] [BY ALL]
```

其中：

```
database_authorities_types ::=
{DBADM | SECADM | BINDADD | CONNECT | CREATETAB | CREATE_EXTERNAL_ROUTINE | CREATE_
NOT_FENCED_ROUTINE | IMPLICIT_SCHEMA | LOAD | QUIESCE_CONNECT}[,...n]
```

参数说明如下。

- ☑ 要撤销 DBADM 权限，必须具有 SYSADM 权限。
- ☑ [,...n]：表示可同时授予多个权限，但同一个权限名称在一个 GRANT 或一个 REVOKE 语句中只能出现一次。
- ☑ 向用户、组授予数据库级权限：可以分别使用 TO USER、TO GROUP 子句指定是向用户还是向组授予数据库级权限。
- ☑ PUBLIC 组：DB2 UDB 在内部只用一个伪组 PUBLIC，它包含一切成功经过身份验证的用户。可以对 PUBLIC 组授予和撤销权限或特权，就像对其他组一样。
- ☑ 如果将权限或特权授予 PUBLIC 组，则任何提供了有效用户 ID 和密码的用户都能够执行 PUBLIC 组有权执行的操作。
- ☑ BY ALL：收回所有显示授权的用户的数据库级权限，无论谁授予他们权限，这是默认选项。

【例 14.6】 用命令为 MRS 组添加 MRKJ 数据库的数据库权限（**实例位置：光盘\TM\sl\14\3**）。

实现步骤如下。

（1）连接 MRKJ 数据库：

```
CONNECT TO mrkj USER mr USING 1;
```

（2）为 MRS 组添加 MRKJ 数据库的数据库权限：

```
GRANT DBADM,CREATETAB,BINDADD,CONNECT,CREATE_NOT_FENCED_ROUTINE,
IMPLICIT_SCHEMA,LOAD,CREATE_EXTERNAL_ROUTINE,QUIESCE_CONNECT
```

```
ON DATABASE
TO GROUP MRS;
```

说明

本例与例 14.5 是等效的。

【例 14.7】 假设 MRS 组拥有在 MRKJ 数据库上的除了 SECADM 以外的所有数据库权限，假设用户 MR 拥有 SYSADM 权限，下面命令是否正确。

```
CONNECT TO mrkj USER mr USING 1;
REVOKE LOAD ON DATABASE FROM GROUP mrs BY ALL;
```

上面的命令试图撤销 MRS 组的 LOAD 权限，但是会报错，且不能执行。

因为假如撤销了 MRS 组的 LOAD 权限，那么它仍然具有 DBADM 权限，而 DBADM 权限隐含了 LOAD 权限，这是矛盾的。

如果要撤销 MRS 组的 LOAD 权限，需要先撤销 MRS 组的 DBADM 权限，然后撤销 MRS 组的 LOAD 权限。

实现步骤如下。

（1）修改数据库权限前首先要连接数据库：

```
CONNECT TO mrkj USER mr USING 1;
```

（2）先撤销 MRS 组的 DBADM 权限：

```
REVOKE DBADM ON DATABASE FROM GROUP MRS BY ALL;
```

（3）然后撤销 MRS 组的 LOAD 权限：

```
REVOKE LOAD ON DATABASE FROM GROUP MRS BY ALL;
```

14.6 特　　权

实例权限级别这种机制用于将一组预先定义的管理权限授予一组用户账号，而特权是明确分配给单独用户或组的，允许他们在数据库对象上执行特定操作（例如，创建和删除索引）。特权严格地定义了用户可以执行的任务。例如，用户可能具有读一个表的数据的特权，但是不能更新此数据。

14.6.1 特权级别

图 14.26 显示了 DB2 UDB 中各种数据库对象（表空间、模式、包、表、索引、视图、例程、序列、XSR 对象、变量）上的特权级别。注意：SYSADM 和 DBADM 权限自动获得实线下面对于某个数据库对象的所有权限和特权。

DB2 UDB 将关于特权的信息存储在 10 个系统编目视图中：

☑ SYSCAT.TBSPACEAUTH：表空间特权。
☑ SYSCAT.SCHEMAAUTH：模式特权。
☑ SYSCAT.PACKAGEAUTH：包特权。
☑ SYSCAT.TABAUTH：表和视图特权。
☑ SYSCAT.COLAUTH：表和视图列特权。
☑ SYSCAT.INDEXAUTH：索引特权。
☑ SYSCAT.ROUTINEAUTH：例程特权。
☑ SYSCAT.SEQUENCEAUTH：序列特权。
☑ SYSCAT.XSROBJECTAUTH：XSR 对象特权。
☑ SYSCAT.VIARIABLEAUTH：变量特权。

可以像查询任何其他视图一样查询这些视图。

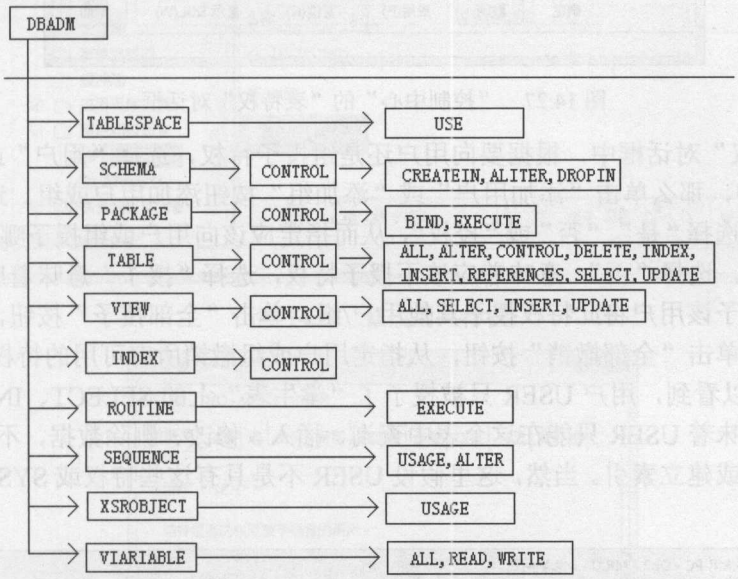


图 14.26　DB2 UDB 特权的层次结构

14.6.2　授予和撤销特权

有两种方式授予和撤销特权。

1. 在“控制中心”修改数据库对象的用户或组范围

【例 14.8】　在“控制中心”修改 MRKJ 数据库中表对象的用户或组范围。

实现步骤如下：

（1）在“控制中心”对象视图中选择“所有系统”→“系统名”→“实例”→“实例名”→“数据库”→“目标数据库名”（如“MRKJ 数据库”），选择包含感兴趣的数据库对象的文件夹（如“表”），在右面的内容窗格中选择感兴趣的数据库对象（如“成绩表”或“学生表”），在该对象上单击鼠标右

键，在弹出的快捷菜单中选择“特权”命令。如果选择 MRKJ 数据库中的“表”文件夹，然后在“学生表”上单击鼠标右键，在弹出的快捷菜单中选择“特权”命令，就打开了如图 14.27 所示的对话框。

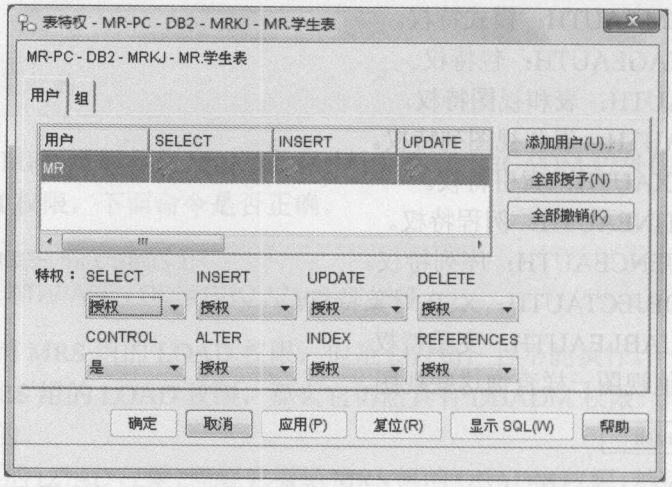


图 14.27 “控制中心”的“表特权”对话框

（2）在“表特权”对话框中，根据要向用户还是组授予特权，选择“用户”或“组”选项卡。如果用户/组不在列表中，那么单击“添加用户”或“添加组”按钮添加用户或组。通过在适当“特权”名称下面的列表框中选择“是”、“否”或“授权”，从而指定应该向用户或组授予哪些特权。选择“是”，意味着应该授予特权，选择“否”，意味着应该不授予特权，选择“授予”意味着应该授予此特权给当前选定的用户，且授予该用户将此特权授予其他用户/组。单击“全部授予”按钮，向指定用户或组授予所有可用的特权。单击“全部撤销”按钮，从指定用户或组撤销所有可用的特权。

在图 14.28 中可以看到，用户 USER 只被授予了“学生表”上的 SELECT、INSERT、UPDATE 和 DELETE 特权，这意味着 USER 只能在这个表中查询、插入、修改、删除数据，不能修改表中的结构，也不能创建外关键字或建立索引。当然，这里假设 USER 不是具有这些特权或 SYSADM/DBADM 权限的组的成员。

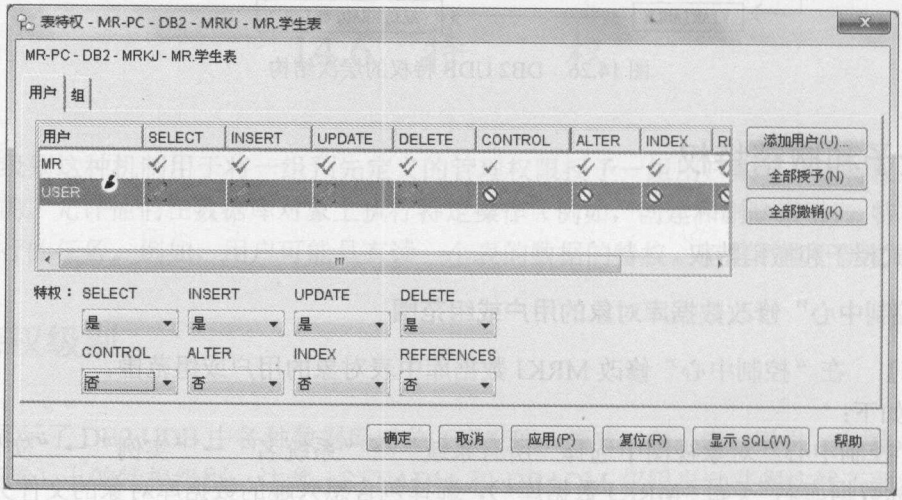


图 14.28 选择授予特权

2. 在“控制中心”修改用户或组拥有的特权

【例 14.9】　在“控制中心”修改用户 USER 拥有的特权。

实现步骤如下：

（1）在“控制中心”对象视图中，选择“所有系统”→“系统名”→“实例”→“实例名”→“数据库”→“数据库名”→“用户和组对象”→“数据库用户”文件夹，如图 14.29 所示。在“控制中心”右面的内容窗格中选择用户“USER”，在 USER 上单击鼠标右键，在弹出的快捷菜单中选择“更改”命令，从而弹出“更改用户”对话框，如图 14.30 所示。

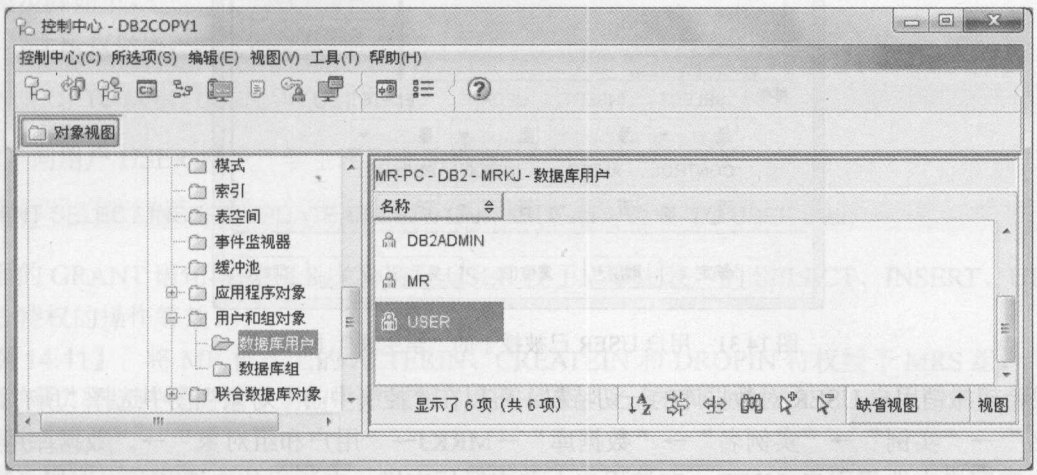


图 14.29　修改用户特权

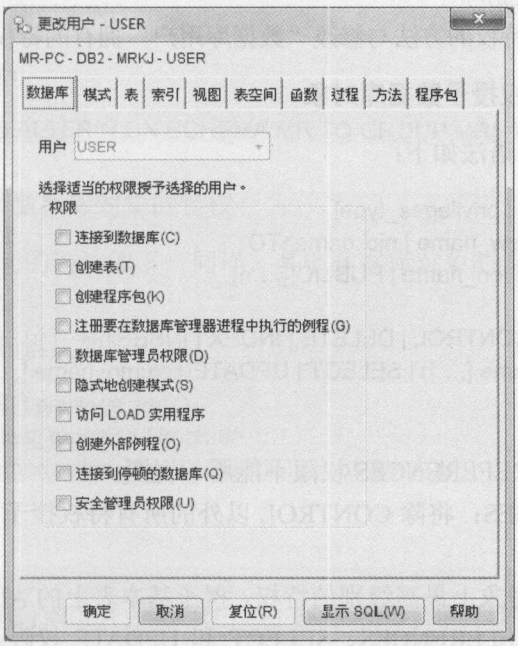


图 14.30　在“控制中心”打开“更改用户”对话框

（2）如果选择“更改用户”对话框中的“表”选项卡，则可以看到例 14.8 中给用户 USER 授予的

“学生表”上的 SELECT、INSERT、UPDATE 和 DELETE 特权，如图 14.31 所示。还可以通过各个数据库对象（如表空间、函数等）选项卡修改 USER 在其他数据库对象上的其他特权。

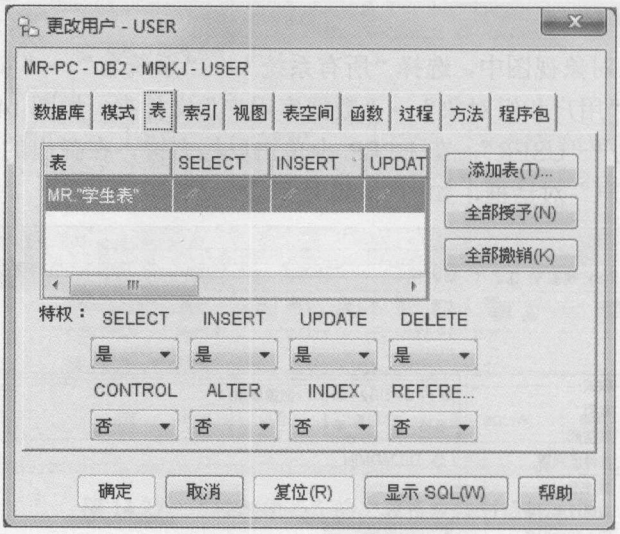


图 14.31　用户 USER 已被授予的“学生表”上的特权

如果希望撤销用户 USER 对数据库的一切特权。可以在“控制中心”对象视图中选择“所有系统”→“系统名”→“实例”→“实例名”→“数据库”→MRKJ→“用户和组对象”→“数据库用户”，在右面的内容窗格中选择用户 USER，在该用户上单击鼠标右键，在弹出的快捷菜单中选择“除去”命令，并在弹出的对话框中确认删除。

修改“数据库组”拥有的特权的方法与修改“数据库用户”拥有的特权的方法相似，这里不再赘述。

3．用 GRANT 语句将特权授予数据库对象

表和视图的 GRANT 语句语法如下：

```
GRANT [ALL [PRIVILEGES] | privileges_type]
ON[TABLE]<table_name | view_name | nickname>TO
{[USER | GROUP ] authorization_name | PUBLIC}[,…n]
[WITH GRANT OPTION]
privileges_type ::=ALTER | CONTROL | DELETE | INDEX | INSERT
| REFERENCES (column_name [,…]) | SELECT | UPDATE(column_name [,…n])
```

参数说明如下。

- ☑ ALTER、INDEX 和 REFERENCES 权限不能用于视图。
- ☑ ALL | ALL PRIVILEGES：将除 CONTROL 以外的所有特权授予 ON 子句中的对象（基本表、视图或昵称）。
- ☑ CONTROL：数据库对象上最高级别的特权。授予基本表上的 ALTER、CONTROL、DELETE、INSERT、INDEX、REFERENCES、SELECT 和 UPDATE 权限；授予视图上的 CONTROL、DELETE、INSERT、SELECT 和 UPDATE 权限，授予昵称上的 ALTER、CONTROL、INDEX 和 REFERENCES 权限；将上述权限（CONTROL 除外）授予其他用户；删除基本表、视图和

昵称。

☑ WITH GRANT OPTION：允许被授权的用户或组的成员将特权授予其他用户、组或角色。不能使用 WITH GRANT OPTION 子句向别的用户或组授予对象的 CONTROL 特权。这种特权必须专门授予用户或组，并只能由具有 SYSADM 或 DBADM 权限的用户执行。WITH GRANT OPTION 只能用于包、例程、模式、表、视图和表空间的 GRANT 语句。

【例 14.10】　用命令向用户 USER 授予“学生表”的 SELECT、INSERT、UPDATE 和 DELETE 特权（**实例位置：光盘\TM\sl\14\4**）。

实现步骤如下。

（1）连接数据库：

```
CONNECT TO MRKJ USER MR USING 1;
```

（2）向用户 USER 授予“学生表”上的特权：

```
GRANT SELECT,INSERT,UPDATE,DELETE ON TABLE mr.学生表 TO USER user;
```

上面的 GRANT 语句与例 14.8 中向用户 USER 授予“学生表”的 SELECT、INSERT、UPDATE、DELETE 特权的操作等效。

【例 14.11】　将 MR 模式上的 ALTERIN、CREATEIN 和 DROPIN 特权授予 MRS 组，同时允许 MRS 组的成员将这些特权授予别的用户或组（**实例位置：光盘\TM\sl\14\5**）。

实现步骤如下。

（1）连接数据库：

```
CONNECT TO MRKJ USER MR USING 1;
```

（2）向 MRS 组授予特权：

```
GRANT CREATEIN,DROPIN,ALTERIN ON SCHEMA MR TO GROUP MRS WITH GRANT OPTION;
```

4．用 REVOKE 语句撤销数据库对象的特权

表和视图的 REVOKE 语句的语法如下。同样，其他数据库对象的 REVOKE 语法也与之类似，请参看 DB2 UDB 文档。

表和视图的 REVOKE 语句语法如下：

```
REVOKE [ALL [PRIVILEGES] | privilege_type]
ON [TABLE]<table_name | view_name | nickname>
FROM {[USER | GROUP] authorization_name | PUBLIC}[,…n] [BY ALL]
```

其中：

```
privilege_type ::=
ALTER | CONTROL | DELETE | INDEX | INSERT | REFERENCES | SELECT | UPDATE
```

参数说明如下。

☑ BY ALL：收回所有显示授权的用户的所有特权，无论谁授予他们特权，这是默认选项。

注意

要撤销数据库对象上的特权，必须具有 DBADM 权限、SYSADM 权限或此对象上的 CONTROL 特权。注意，拥有 WITH GRANT OPTION 特权并不足以撤销这一特权。要从另一个用户撤销 CONTROL 特权，必须具有 SYSADM 或 DBADM 权限。

从用户或组撤销特权时将撤销其他任何账号授予它的特权。但是从用户或组撤销特权并不撤销这个用户/组授予别的账号的同一特权。例如，假设用户 NBH 将 SELECT WITH GRANT OPTION 授予用户 RICK，然后 RICK 将 SELECT 授予用户 RAVI 和 CHRIS。如果以后 NBH 从 RICK 撤销了 SELECT 特权，那么 RAVI 和 CHRIS 仍然拥有 SELECT 特权。

【例 14.12】 撤销用户 USER 在“学生表”上的所有特权（**实例位置：光盘\TM\sl\14\6**）。

实现步骤如下。

（1）连接数据库：

```
CONNECT TO MRKJ USER MR USING 1;
```

（2）撤销用户 USER 在“学生表”上的所有特权：

```
REVOKE ALL PRIVILEGES ON TABLE mr.学生表 FROM USER user;
```

5. 从组中的一个成员撤销特定特权

如果希望将一种特权授予一个组，然后只从这个组中的一个成员撤销这一特权，但是，不能撤销并未显示授予的特权。在这种情况下可以从这个组中删除这个成员或者创建一个只包含组中其他成员的新组，并将特权授予这个新组。

14.6.3 隐式特权

在某些情况下，当用户创建一个数据库对象（如表或包）时，或者授予 DBADM 权限级别时，数据库管理器会隐式地将某些特权授予用户。重要的是，要了解授予了哪些隐式特权，以及这些隐式特权的安全影响。表 14.3 总结了授予隐式特权的情况。

表 14.3 对于不同操作授予的隐式特权的总结

操　作	向执行此操作的用户授予的隐式特权
创建新的数据库	将 DBADM 权限以及 BINDADD、CONNECT、CREATETAB、CREATE_EXTERNAL、ROUTINE、CREATE_NOT_FENCED_ROUTNE、IMPLICIT_SCHEMA、LOAD 和 QUIESCE_CONNECT 权限授予创建者（SYSADM 或 SYSCTRL）； 将 BINDADD、CREATETAB、CONNECT 和 IMPLICIT_SCHEMA 授予 PUBLIC； 将每个成功绑定的实用程序上的 BIND 和 EXECUTE 特权授予 PUBLIC； 将系统编目表和视图上的 SELECT 授予 PUBLIC； 将 USERSPACE1 表空间上的 USE 特权授予 PUBLIC； 将 SYSFUN 模式中的所有函数上的 EXECUTE WITH GRANT 特权授予 PUBLIC； 将 SYSIBM 模式中的所有过程上的 EXECUTE 特权授予 PUBLIC

续表

操　作	向执行此操作的用户授予的隐式特权
授予 DBADM 权限	授予 BINDADD、CONNECT、CREATETAB、CREATE_EXTERNAL_ROUTINE、CREATE_NOT_FENCED_ROUTINE、IMPLICIT_SCHEMA、LOAD 和 QUIESCE_CONNECT
模式	在明确创建时，将 CREATEIN、ALTERIN、DROPIN 授予创建这个模式的用户； 在隐式创建时，将 CREATEIN 授予 PUBLIC
创建对象（表、索引、包）	将 CONTROL 授予对象创建者
创建视图	只有在用户拥有视图定义中引用的所有表、视图和别名的 CONTROL 特权时，授予 CONTROL 特权

注意

实例级权限级别(SYSADM、SYSCTRL、SYSMAINT 和 SYSMON)和组成员关系是在 DB2 UDB 之外定义的，因此不会反映在系统编目表中。

【例 14.13】 假设已经将 DBADM 权限授予 MRS 组，撤销此权限（**实例位置：光盘\TM\sl\14\7**）。实现步骤如下。

（1）连接数据库：

```
CONNECT TO MRKJ USER MR USING 1;
```

（2）撤销 MRS 组的 DBADM 权限：

```
GRANT DBADM ON DATABASE TO GROUP mrs;
```

只有拥有 SYSADM 权限的用户才能撤销 DBADM 权限。

在执行上面的命令之后，MRS 组不再拥有 DBADM 权限，但是，它仍然拥有数据库上的 CREATETAB、BINDADD、CONNECT、CREATE_NOT_FENCED_ROUTINE、IMPLICIT_SCHEMA、LOAD、CREATE_EXTERNAL_ROUTINE、QUIESCE_CONNECT 权限，这些权限是原来将 DBADM 权限授予 MRS 组时隐式授予的。需要从 MRS 组明确地撤销这些权限。

注意

一种好的安全实践是，在创建新数据库之后，明确地撤销隐式授予 PUBLIC 的许多特权，这可以保护系统，确保只有应该访问数据库的用户才能这么做。

14.7　LBAC 凭证

基于标签的访问控制（LBAC）是 DB2 从 V9 版本开始引入的一种对数据访问的用户进行安全控制的功能。LBAC 使操作者能够准确地确定对于各行各列具有写访问权的用户和具有读访问权的用户。

LBAC 功能的配置并不复杂，所有 LBAC 中的安全配置工作都由安全管理员（SECADM）执行。

要对表中的数据实现 LBAC 的安全功能，首先安全管理员要依次创建 3 种数据库对象：安全标签组件、安全策略和安全标签，其中每个安全标签都有“高”或“低”信任级别。然后对表应用安全策略，同时将安全标签与表中的行和列进行关联，和安全标签相关联的数据称为受保护的数据。接下来再将不同的安全标签分配给不同的用户，当用户尝试访问受保护数据时，首先会把该用户的安全标签和用于保护该数据的安全标签进行信任级别比较。如果用户拥有的安全标签的信任级别高于受保护数据的安全标签的信任级别，那么此用户可以访问该数据，反之，用户的操作将被阻止。安全管理员还可以将豁免权赋予用户。豁免权允许用户访问安全标签不允许访问的受保护数据。用户所拥有的安全标签和豁免权统称为 LBAC 凭证。如果要访问 LBAC 凭证不允许访问的受保护列，访问就会失败，并且会发出错误消息。如果要读取 LBAC 凭证不允许读取的受保护行，则 DB2 会将那些行视为不存在。

14.8 GET AUTHORIZATIONS 命令

GET AUTHORIZATIONS 命令是一个报告当前用户的特权的便捷命令。这个命令使用在数据库管理器配置文件和授权系统编目视图（SYSCAT.DBAUTH）中获取的值。

【例 14.14】 查看用户 MR 的所有权限（**实例位置：光盘\TM\sl\14\8**）。

```
CONNECT TO MRKJ USER MR USING 1;
GET AUTHORIZATIONS;
```

执行结果如图 14.32 所示。

```
当前用户的管理权限

直接 SYSADM 权限                = NO
直接 SYSCTRL 权限               = NO
直接 SYSMAINT 权限              = NO
直接 DBADM 权限                 = YES
直接 CREATETAB 权限             = NO
直接 BINDADD 权限               = NO
直接 CONNECT 权限               = NO
直接 CREATE_NOT_FENC 权限       = NO
直接 IMPLICIT_SCHEMA 权限       = NO
直接 LOAD 权限                  = NO
直接 QUIESCE_CONNECT 权限       = NO
直接 CREATE_EXTERNAL_ROUTINE 权限  = NO
直接 SYSMON 权限                = NO

间接 SYSADM 权限                = YES
间接 SYSCTRL 权限               = NO
间接 SYSMAINT 权限              = NO
间接 DBADM 权限                 = NO
```

图 14.32 查看用户所有权限

“直接”权限意味着此权限是明确授予此用户的，“间接”权限意味着此用户属于具有此权限的组。如果用户被明确授予了此权限，同时又属于具有此权限的组，那么他同时具有直接权限和间接权限。

14.9　使用模式控制对数据库对象的访问

新的 DB2 UDB 数据库管理员经常会问的一个问题是，如何为用户创建适当的环境，让他们能够创建和删除自己拥有的数据库对象，同时限制其他用户访问这些对象。给每个用户提供一个他自己专用的物理数据库可以解决这个问题，但这不是优化的解决方案。更好的解决方案可能是通过使用模式控制数据库对象访问。

模式是一种数据库对象，用于按照逻辑将相关的数据库对象分组，还常常用来表示对象所属权。模式具有相关联的特权，使模式所有者能够控制哪些用户有权在这个模式中创建、修改和删除对象。模式所有者最初具有这个模式上的所有特权，并能够将这些特权授予其他人。具有 SYSADM 或 DBADM 权限的用户可以修改用户在任何模式上拥有的特权。

在默认情况下，在创建数据库时，所有用户具有 IMPLICIT_SCHEMA 权限。这使任何用户可以在任何尚不存在的模式中创建对象。隐式创建的模式允许任何用户在其中创建其他对象。如果从 PUBLIC 撤销了 IMPLICIT_SCHEMA 权限，那么可以使用 CREATE SCHEMA 语句明确地创建模式，而且已经被隐式授予 IMPLICIT_SCHEMA 权限的用户（例如具有 DBADM 权限的用户）仍然可以隐式地创建模式。

为了让每个用户可以控制自己的数据库对象，数据库管理员可以为每个用户明确地创建一个模式，然后，管理员根据需要将模式上的特权授予单独的用户。这样，就可以防止其他用户篡改在这个模式中创建的任何对象。为了进一步保护系统，还可以从 PUBLIC 撤销 IMPLICIT_SCHEMA 权限，这样的话，如果用户想创建数据库对象，那么必须通过他具有适当特权的模式来创建。这种方式使用户能够创建他们需要的数据库对象，同时避免他们篡改其他用户创建的数据库对象，或者将自己和其他人创建的对象混在一起。

14.10　小　　结

本章主要介绍了 DB2 身份验证、管理权限和特权的基本操作。在访问 DB2 数据库时，为了确保数据库的安全性，用户必须输入用户名和密码后，才能连接数据库。另外，为了防止合法用户的非法访问，DB2 提供了管理权限和特权，以防止用户对数据库的非法操作。

14.11　实践与练习

（1）什么是认证？DB2 支持的认证有哪些？（**答案位置：光盘\TM\sl\14\9**）

（2）DB2 有哪几种身份验证？（**答案位置：光盘\TM\sl\14\10**）

（3）什么是授权？DB2 使用什么机制来判断授权是否成功？（**答案位置：光盘\TM\sl\14\11**）

（4）列举出 3 个数据库级权限并说出其用途。（**答案位置：光盘\TM\sl\14\12**）

第15章

数据库的备份与恢复

（视频讲解：53分钟）

在以IT架构为基础的现代企业环境中，数据是企业最宝贵的资源。数据的丢失往往意味着难以计量和弥补的损失。而在数据库中，保证数据不丢失的最后一道屏障往往是数据库的备份。为了更好地保护客户的关键数据，DB2数据库提供了健全的备份、恢复功能，以方便用户制订并实施满足自己业务需求的数据保护策略。

通过阅读本章，您可以：

- 了解数据库的备份与恢复基础知识
- 掌握事务日志记录
- 掌握如何进行数据备份
- 熟悉如何进行数据恢复
- 熟悉常见的恢复场景

15.1　备份与恢复

在信息日趋发达的时代，数据显得尤其重要。任何数据的丢失都可能对企业造成难以挽回的损失。因此，为保证数据的可用性、可恢复性，在存储级、系统级和数据库、应用级都涌现出很多技术。如 RAID 技术通过磁盘数据冗余尽量减少磁盘损坏造成的数据丢失，HA 技术通过双机确保数据的持续可用。尽管以上技术可以最大程度地保证系统的持续可用，但并不能解决数据人为误删、软件故障、电源故障、硬件故障等造成的数据损失。DB2 数据库提供了备份与恢复机制，从而可以使用户放心地使用。其中，备份是将数据信息保存起来，恢复是将原来备份的数据信息还原到数据库中。

数据库没有备份，会出现下列问题：

（1）由于硬件原因引发的故障。包括电源和其他硬件在内的故障都可能导致数据库处于不一致状态。

（2）存储介质引起的故障，如磁盘阵列、磁盘介质故障等都可能造成数据丢失。磁盘有一定的生命周期，而一些老旧磁盘，出问题的概率很大。

（3）人为操作故障，如不小心删了数据、移除了表、删除了文件系统等。数据库管理员经常处于高压和加班的疲惫状态，误操作很常见，但不经意间的错误可能造成难以挽回的损失，因此操作前一定要谨慎。

（4）战争和自然灾害等。如我国的“5 • 12”汶川大地震造成多家企业数据中心瘫痪。

作为 DBA，需要根据企业的应用类型和特点，定制有效的备份恢复策略，从以下 3 个方面确保数据库的可恢复性：

☑ 将数据库的故障率降到最低，从而使数据库保持最大的可用性。

☑ 当数据库发生故障后，尽量减少数据丢失，从而使数据具有最大的可恢复性。

☑ 当数据库发生故障后，将恢复时间减到最少，从而使恢复的效率达到最高。

为了了解数据库备份/恢复相关的基本概念，我们介绍两个最常见的场景。第一个场景是某 DBA 凌晨 2 点对数据库做了备份，随后进行了一些业务处理；第二天上午 10 点用户不小心破坏了系统数据，那么如何进行数据恢复呢？对于采用了归档模式的数据库，采用的方法是恢复+前滚，即首先恢复凌晨 2 点的备份介质，然后利用日志前滚到 10 点出事前的时间点。

第二个场景是数据库正在执行过程中突然断电。这时某些数据可能还未写到数据盘上，那么 DB2 如何保证数据库的一致性呢？答案是通过 DB2 事务日志和崩溃恢复。

15.2　恢复操作的种类

DB2 数据库中主要包括 3 种恢复操作：版本恢复、前滚恢复和崩溃恢复，下面分别进行介绍。

☑ 版本恢复（Restore）：每个备份相当于一个版本，版本恢复是将数据库恢复到某个备份时的状态，这个备份之后的数据将会丢失。在第一个场景中，将数据恢复到凌晨 2 点的操作就是版

本恢复。

- ☑ 前滚恢复（Rollforweard）：前滚恢复是在版本恢复的基础上进行的日志重用。前滚恢复要求日志必须是归档类型，可以前滚日志到某个时间点，也可以到日志文件的结尾。如果需要追回 2 点以后的操作数据，则需要在版本恢复的基础上，将日志前滚到 2 点以后的某个时间点，前滚操作的机制是利用日志尽量减少数据的丢失。注意：前滚和回滚是完全不同的概念，不要混淆。
- ☑ 崩溃恢复（Crash Recovery）：崩溃恢复的目的是保证数据库的一致性。在第二个场景中假定执行完事务的第一条语句后，机器突然异常停机，这时操作还没有结束，事务就被中断了，这时数据库会处于不一致状态。当重新启动系统后，必须对数据库做崩溃恢复，即回滚（Undo）还没有提交的数据，并重做（Redo）已经提交但还没有写入磁盘的数据，将数据库恢复到一致状态。默认情况下，崩溃恢复是自动执行的，不需要人工干预。

15.3 事务日志记录

任何数据库管理系统都必须拥有确保数据一致性和可恢复性的机制，DB2 数据库系统为确保那些非常重要的特性所使用的众多机制之一是事务日志记录。

15.3.1 为什么需要日志

首先，假设您是一个数据库系统的设计师，为了维护数据库的一致性，您将会用什么机制来避免下面的问题呢？

（1）当数据库崩溃恢复时，原先写在缓冲池中的数据可能没有来得及写入磁盘，这样磁盘的数据还是修改之前的记录。

（2）当操作回滚的时候，我们需要知道被修改的数据。

（3）数据库被恢复版本之后，我们需要全部历史操作将数据库前滚到一个给定的时间点。

从这几个问题中我们知道，至少数据库需要一种机制，能够记录所有的数据更新操作及被更新之前的旧数据。同时，该机制必须保证记录在真正的数据到达磁盘前被写入永久存储系统，以便在系统以外崩溃时能够恢复到一致的状态。

在 DB2 中，这种机制使用日志功能实现。可以把日志认为是在一条事务被落实之前，能够保证其记录写入永久存储系统的一种方法。

可能有的读者会问：为什么需要日志呢？我可以直接把编好的数据写入磁盘，这样不是更好吗？之所以需要日志，主要是从性能上考虑。通常情况下，每一个事务（事务的详细概念见第 19 章）包含若干条数据更改语句，每个更改可能需要操作大量的数据。如果将这些数据直接写入磁盘才返回给应用端，那么将严重影响写的性能，所以，DB2 采用写日志优先算法，也就是先写日志，再写数据，而写数据的过程是异步的。

图 15.1 是 DB2 日志原理图，当插入、更改、删除数据时，该条记录并不会直接写到数据盘，而是

首先写到日志文件。而缓冲池数据写到数据盘是个异步过程。这样即使某一时刻突然宕机，数据没有从缓冲池内存写到数据盘也没有关系，因为数据已经记到日志文件里面了。当重新开机进行恢复时，DB2 将日志文件的内容重写到数据盘，保证数据库的一致性。

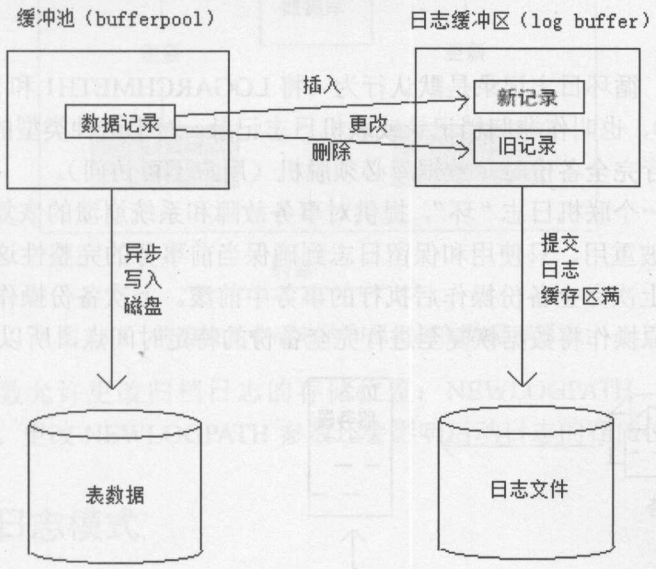


图 15.1　DB2 日志原理

15.3.2　事务日志记录的概念

事务日志记录是一个与数据库文件分开的文件，它存储对数据库进行的所有更改，并全部记录插入、更新、删除、提交、回滚和数据库模式变化。事务日志记录还称作前滚日志记录或重做日志记录。

事务日志的意义在于数据恢复，DB2 实施了提前写日志存档模式，当发出删除、插入或更新数据库中某一条数据的 SQL 调用时，所作出的数据变更首先要写到日志中去。当发出一条 SQL 提交确认命令时，DB2 首先要把所需要的日志写入磁盘。如果断电的话，所有被提交的事务都会重新做一遍，而未提交的事务则会回滚。所以说，管理好事务日志文件，对于保障数据库中数据的安全非常重要。

默认情况下，所有数据库都使用事务日志。事务日志的使用是可选的，但是，除非因特殊原因而不使用，否则应始终使用它。运行带有事务日志的数据库可提供更强的故障保护功能、更好的性能以及数据复制功能。

15.3.3　主日志文件和辅助日志文件

DB2 数据库的日志文件分为主日志文件和辅助日志文件，其中主日志文件在第一个连接到达数据库或者数据库被激活后立即分配，默认情况下会为每个数据库创建固定的 3 个主日志文件。当主日志文件已满时，一次动态分配一个辅助日志文件，默认情况下最多可分配两个辅助日志文件。用户可以调整主日志文件和辅助日志文件的数目。

15.3.4 循环日志记录和归档日志记录

1. 循环日志记录

当创建新数据库时，循环日志记录是默认行为（将 LOGARCHMETH1 和 LOGARCHMETH2 数据库配置参数设置为 OFF），也叫作非归档记录或联机日志记录。对于这种类型的日志记录，只允许完整的脱机数据库备份。进行完全备份时，数据库必须脱机（用户不可访问）。

循环日志记录使用一个联机日志“环”，提供对事务故障和系统崩溃的恢复，如图 15.2 所示。当事务提交后，日志文件会被重用。只使用和保留日志到确保当前事务的完整性这样一个程度。循环日志记录不允许将数据库在上次完全备份操作后执行的事务中前滚。上次备份操作后发生的所有更改都将丢失，因为这种类型复原操作将数据恢复至进行完全备份的特定时间点，所以它称为版本恢复。

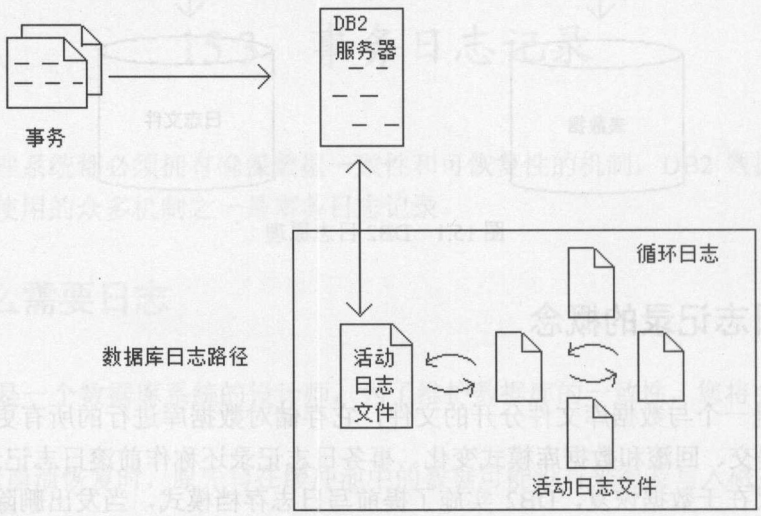


图 15.2 循环日志记录

在崩溃恢复期间，使用活动日志来防止故障（系统电源或应用程序错误）使数据库处于不一致的状态。活动日志位于数据库日志路径目录中。

2. 归档日志记录

归档日志记录是针对前滚恢复使用的。归档日志文件永远都不可重用（即覆盖），当存储于某个日志文件中的所有记录都不再需要用于恢复时，该日志文件将被标记为非活动而不是可重用。归档日志是曾经活动的，但不再是崩溃恢复所必需的日志。使用 LOGARCHMETH1 数据库配置参数来启用归档日志记录。图 15.3 显示了前滚恢复中活动的和已归档的数据库日志。运行时间较长的事务中，可以有多个活动日志。

只有当将数据库配置为归档日志记录时，才支持联机备份。在联机备份操作期间，将记录对数据库的所有活动。复原联机备份映像时，必须至少将日志前滚至完成备份操作的时间点。至此，日志在复原数据库时必须已归档并可用。在联机备份完成后，DB2 将强制关闭当前活动日志，从而使日志归

档。这样就可确保联机备份有一组完整的归档日志可用于恢复。

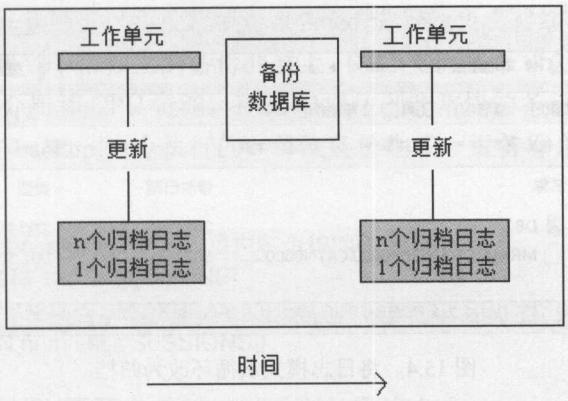


图 15.3　前滚恢复中活动的和已归档的数据库日志

以下数据库配置参数允许更改归档日志的存储位置：NEWLOGPATH、LOGARCHMETH1 和 LOGARCHMETH2 参数。更改 NEWLOGPATH 参数还会影响活动日志的存储位置。

15.3.5　修改事务日志模式

【例 15.1】　将 MRKJ 数据库日志模式从循环修改为归档或者从归档修改为循环（**实例位置：光盘\TM\sl\15\1**）。

执行以下命令可以将日志模式从循环修改为归档。

实现步骤如下。

（1）在磁盘上创建一个文件夹（例如，D:\mrkj\archive），磁盘上必须有足够的空间存储归档日志文件。保证归档文件目标文件夹与活动日志文件目标文件夹分开。

（2）用以下命令明确终止命令行处理器的后端进程：

```
TERMINATE;
```

（3）指定归档日志文件目标文件夹，从而打开归档日志模式。

```
UPDATE DB CFG FOR mrkj USING LOGARCHMETH1 "DISK:d:\mrkj\archive";
```

（4）用以下命令明确终止命令行处理器的后端进程：

```
TERMINATE;
```

（5）迫使本地或远程用户或应用程序断开 DB2 系统，以便维护 DB2 服务器。

```
FORCE APPLICATION ALL;
```

（6）再用以下命令完全备份数据库后，归档日志模式才会生效。

```
BACKUP DATABASE mrkj TO d:\mrkj\database;
    --将 MRKJ 数据库脱机完全备份到 d:\mrkj\database 目录下
```

本例运行结果如图 15.4 所示。

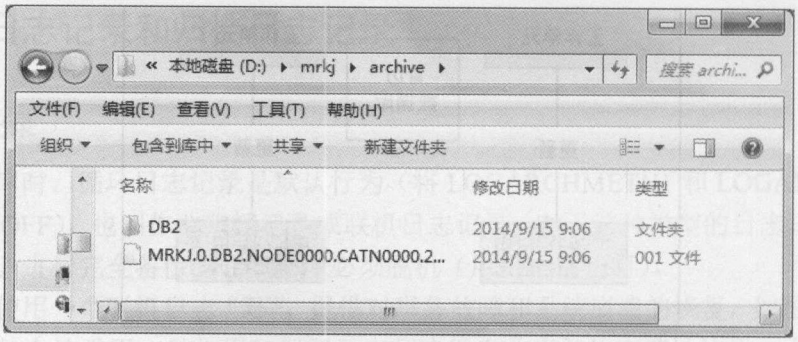


图 15.4 将日志模式从循环改为归档

执行以下命令可以将日志模式从归档改为循环：

```
CONNECT TO mrkj USER mr USING 1;
UPDATE DATABASE CONFIGURATION USING LOGARCHMETH1 OFF IMMEDIATE;
```

15.4 备 份

做好数据备份是对数据库维护人员的最基本要求，如果没有备份，那么当出现各类故障或人为操作失误时，数据库中的数据就将面临很大风险。

所以，DBA 必须根据应用特点，指定有效的备份策略，并付诸实施。备份策略需要考虑很多方面，如成本、业务类型、对数据可用性的需求、对恢复数据长度的需求、运维管理的复杂度等。对于交易系统，为了保证对数据的持续可用性，建议采用在线备份方式，如果数据库不大，可考虑数据库全备份，相对于增量备份来说，该方式恢复起来简单，如果数据库很大，可考虑每周做一次全备份，每天做增量。对于仓库类系统，由于数据量巨大，很多客户无法提供如此庞大的存储空间，也无法忍受长时间的业务中断，这时可考虑采用数据导出的方法将一些重要的表数据备份。

以下是从各个角度对备份进行的分类：

- ☑ 根据备份时是否允许业务访问可分为离线备份和在线备份。离线备份（offline backup）要求备份时没有任何应用连接，而在线备份（online backup）则允许在不中断业务的情况下进行备份，备份时允许读写数据。
- ☑ 根据备份的对象可分为数据库备份和表空间备份。数据库备份是对整个数据库所有数据所做的备份，表空间备份则是针对数据库中某个或某些表空间做的备份，一般是经常变化而又比较重要的表空间。
- ☑ 根据备份的规模可分为完全备份和增量备份。完全备份是对库或表空间所有数据所做的备份，而增量备份只备份前一次备份以来变化的数据。

DB2 的备份采用的是逻辑备份方式，也就是基于页备份，DB2 会备份每个数据页和索引页，而不是备份物理文件，好处是在恢复时可以更改磁盘布局，更加灵活方便。

DB2 备份是高度并行的，通过几组进程协调完成：db2bm 将数据从磁盘读入备份缓冲区（有时会使用 db2pfchr 异步读取），一旦缓冲区页面已满，db2med 进程就负责将缓冲区数据写到备份介质。每个 db2med 进程可以将数据写到一个独立的备份介质上，DB2 支持多个 db2med 线程同时写入多个备份介质。所以，有足够的备份介质和 CPU 资源，DB2 可以达到非常快的备份速度。

DB2 提供的备份是通过 backup 命令执行的，每次备份生成一组备份介质。下面看一下 backup 命令的语法：

```
BACKUP DATABASE database-alias [USER 用户名 [USING 密码]]
[TABLESPACE (<表空间名称 1>, ...)] [ONLINE]
[INCREMENTAL [DELTA]] [USE <TSM | XBSA> [OPEN num-sess SESSIONS]] |
[LOAD library-name [OPEN num-sess SESSIONS]]
TO dir/dev
[WITH num-buff BUFFERS] [BUFFER buffer-size] [PARALLELISM n]
[COMPRESS <comprlib-name>] [EXCLUDE]
[UTIL_IMPACT_PRIORITY<priority>][INCLUDE LOGS | EXCLUDE LOGS]
[WITHOUT PROMPTING]
```

参数说明如下。

- ☑ ONLINE：指定在线备份，默认的情况下是离线备份。只有当日志是归档日志模式下，才允许在线备份。
- ☑ Tablespace：指定对某个或某些表空间备份，在归档模式下才可用。
- ☑ Incremental：指定增量备份。增量备份是在最近一次完全备份以来变化数据的备份。Incremental delta 备份是在最近一次备份（任何类型）以来变化数据库的备份。
- ☑ With、Buffer、Parallelism：这几个参数是用来提升性能的，DB2 会自动选择优化这些参数。
- ☑ Compress：压缩备份介质。这个对于大数据库来说，指定 Compress 选项可以大大降低备份介质空间，但 Compress 选项需要消耗额外的 CPU 时间。
- ☑ Include logs：这个选项只适用于在线备份（online backup）时，当数据库备份结束时，会将当前的活动日志归档，并将其打包到备份介质中。

15.4.1　离线备份

离线备份（offline backup）要求对数据库的所有连接断开，数据库是不可用的。由于备份过程中没有日志产生，因此离线备份数据和数据库总是一致的。

离线备份的语法如下：

```
backup database <db> [to <path>]
```

15.4.2　在线备份

在线备份（online backup）是指备份过程中数据库仍然是可读可写的。为了保持一致性，在线备份要求日志模式必须是归档的，这样可以保证在备份过程中，变化的数据可以通过日志保留。

在线备份的语法如下：

```
backup database <db> online [to <path>] [include logs]
```

Include logs 代表什么含义呢？相信很多人对此有疑问。对于在线备份，从开始备份到结束备份期间，对数据库所做的增、删、改操作都将记录到日志中。假如这时要在另外一台机器上进行恢复，那么除了要拷贝备份介质外，还要将备份期间产生的日志复制到目标端并进行前滚，才能确保数据库恢复成功。但是这种方法比较繁琐，因为 DBA 需要单独传送日志文件，并且需要找出哪些是必须传送的日志文件。Include logs 选项就是这个目的，当在线备份结束时，DB2 会关闭当前活动的日志，并进行归档，然后将其打包保存到备份介质中，这样只需要传送一个备份介质就可以满足最小恢复的要求。这个选项是可选的，但强烈建议加上。

【例 15.2】 将 MRKJ 数据库备份到 D:\mrkj 文件夹中（**实例位置：光盘\TM\sl\15\2**）。

```
backup database mrkj online to D:\mrkj include logs;
```

本例运行结果如图 15.5 所示。

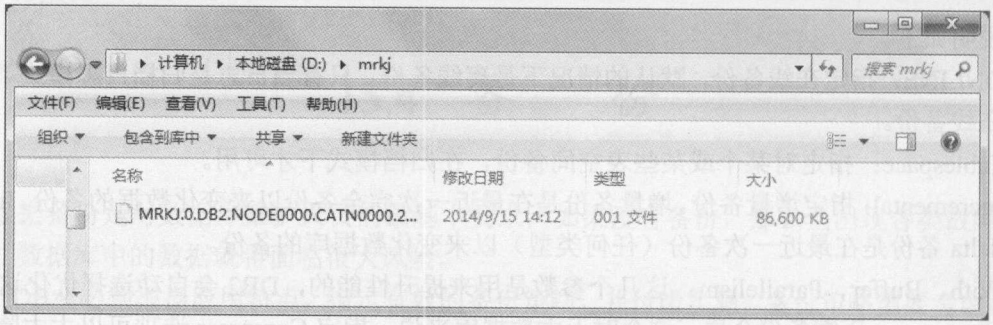


图 15.5 在线备份

注意

在线备份要求日志是归档模式，在线备份时最好不要执行其他的运维任务，例如 runstats 等。否则可能会出现因锁竞争而回滚的情况。

15.4.3 表空间备份

对于较小的数据库，建议采用数据库备份，因为恢复比较简单。但对于大容量数据库来说，全库备份需要的存储空间太大，备份时间也较长，这时可选择对某些重要的表空间做备份。

对表空间进行备份时，建议将相关联的表空间同时备份，例如表 T1 在 TS1 表空间，T2 在 TS2 表空间，假如表 T1 和 T2 相关，则最好将 TS1 和 TS2 同时备份，否则可能在恢复时遇到数据不一致的情况。

表空间备份的语法如下：

```
backup database <db> tablespace (<tbs1>[,<tbs2>]) online [to <path>]
```

15.4.4　增量备份

对于大容量数据库，每次全备份需要占用大量的时间和空间，这时可考虑做增量备份，增量备份的优点表现在：备份介质占用空间更小，备份时间更快，对系统的影响更小，恢复时需要前滚的日志可能更少，但增量备份在恢复时相对复杂。

DB2 支持两种增量备份：累积备份（cumulative backup）和迭代增量备份（delta backup）。累计备份是在全备分的基础上做的备份，而迭代增量则是在一次备份的基础上做的备份，上一次备份可以是任意形式的备份。图 15.6 演示了累积备份策略，在这个策略中，每周日做全备份，从周一到周六做增量备份，每次备份都是在周日全备的基础上累积的。

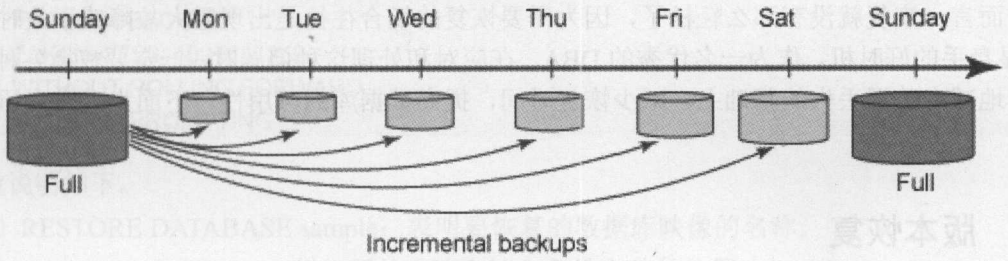


图 15.6　累积增量备份

如图 15.7 所示是迭代增量备份，在这个策略中，每周日做全备份，从周一到周六做迭代增量，每次都是在前一天的基础上的增量。

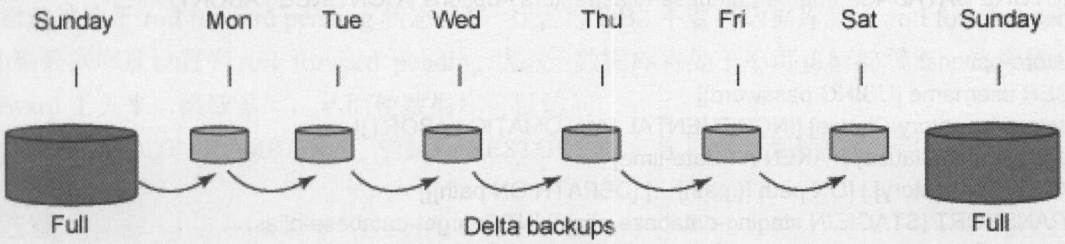


图 15.7　迭代增量备份

增量备份的语法如下：

```
backup database <db> [tablespace (<tbs1>[,<tbs2>])] [online] incremental [delta] [to <path>]
```

参数说明如下。

☑ incremental：表示累积增量备份。

☑ incremental delta：表示迭代增量备份。

DB2 增量备份的步骤如下：

（1）更改数据库参数，使之支持增量备份。

```
update database cfg using TRACKMOD on;
```

（2）在执行增量备份前，必须完全离线备份数据库一次。

```
backup database <db> to [path];
```

（3）执行增量备份命令。

```
backup database <db> online incremental to [path];
```

15.5 恢 复

DB2 支持 3 种类型的恢复：版本恢复、前滚恢复和崩溃恢复（在本章 15.2 节中已经介绍过）。相对于备份而言，恢复就没有那么轻松了，因为需要恢复的场合往往是出现重大故障或事故时，这也是 DBA 大显身手的好时机。作为一名优秀的 DBA，在应对和处理这种问题时，一定要权衡各种方案，在最大程度地减少数据丢失的基础上，减少恢复时间，提高数据库的可用性。下面为大家介绍几种恢复方案。

15.5.1 版本恢复

DB2 的每个备份都是一个版本，版本恢复就是将数据恢复到某个备份时的状态，是备份的逆向过程。版本恢复通过 Restore 命令实现，首先看一下 Restore 命令的语法：

```
RESTORE DATABASE source-database-alias{restore –options | CONTINUE | ABORT}

Restore-options:
[USER username [USING password]]
[Restore-Inventory-Clause] [INCREMENTAL [AUTOMATIC | ABORT]]
[Media-Target-Clause] [TAKEN AT date-time]
[[TO target-directory] | [ON path [{,path}…] [DBPATH ON path]]]
[[TRANSPORT [STAGE IN staging-database-alias]] INTO target-database-alias]
[LOGTARGET {directory | {{INCLUDE | EXCLUDE} [FORCE]}}]
[NEWLOGPATH directory] [WITH num-buff  BUFFERS] [BUFFER buffer-size]
[REPLACE HISTORY FILE] [REPLACE EXISTING]
[REDIRECT [GENERATE SCRIPT file-name]] [PARALLELISM n] [COMPRLIB lib-name]
[COMPROPTS options-string] [WITHOUT ROLLING FORWARD]
[WITHOUT PROMPTING]

Restore-Inventory-Clause:
Rebuild-options | TABLESPACE[ONLINE] |
TABLESPACE (tblspace-name [{,tblspace-name}…])
[SCHEMA (schema-name [{,schema-name}…])] [ONLINE] |
HISTORY FILE [ONLINE] |
LOGS [ONLINE] |
COMPRESSION LIBRARY [ONLINE]
```

```
Media-Target-Clause:
USE {TSM | XBSA} [OPEN num-sess SESSIONS] | SNAPSHOT [LIBRARY shared-lib]
[OPTIONS {options-string | options-filename}] |
LOAD lib-name [OPEN num-sess SESSIONS]
[options {options-string | options-filename}]
FROM dir/dev [{,dir/dev}]

Rebuild-options:
REBUILD WITH {ALL TABLESPACES IN {DATABASE | IMAGE} [EXCEPT TABLESPACE
(tblspace-name [{,tblspace-name}...])]} | TABLESPACE (tblspace-name [{,tblspace-name}...])
```

【例 15.3】　对 sample 数据库进行恢复。

```
(1) RESTORE DATABASE sample
(2) FROM /DBBACKUP
(3) TAKEN AT 2014091631258
(4) WITHOUT ROLLING FORWARD
(5) WITHOUT PROMPTING
```

参数说明如下。

（1）RESTORE DATABASE sample：表明要恢复的数据库映像的名称。

（2）FROM /DBBACKUP：指定要从中读取输入备份文件的位置。

（3）TAEN AT 2014091631258：如果目录中有多个备份映像，该选项将基于时间戳（备份名称的一部分）标识特定的备份。

（4）WITHOUT ROLLING FORWARD：如果数据库启用了归档日志记录，那么当数据库被恢复时，将自动被置于 roll forward pending 状态。这一行告诉 DB2 不要将数据库置于 roll forward pending 状态。如果数据库被设置为 roll forward pending 状态，数据库将处于不可访问的状态，这样就必须使用 roll forward 工具来前滚数据库，从而使数据库可以访问。

（5）WITHOUT PROMPTING：当执行 RESTORE 时，你将看不到任何提示。

注意

以上语法中没有关键字 OFFLINE，因为这是默认模式。对于 RESTORE 实用程序来说，这是能用于数据库的唯一模式。

在以下情况中，不能使用 WIHOUT ROLLING FORWARD 选项：

☑ 从在线备份映像中复原。

☑ 从表空间级的备份映像中复原。

如果备份映像中包括日志文件，那么可以使用 RESTORE DATABASE 命令的 LOGTARGET 选项恢复日志文件。

1. 表空间恢复

可以使用完整数据库备份或表空间备份来恢复表空间。对于表空间恢复，要小心地做好计划，因

为稍有不慎就会使数据处于不一致状态。

【例 15.4】 表空间 RESTORE 命令。

```
（1）RESTORE DATABASE sample
（2）TABLESPACE(mytblspace1)
（3）ONLINE
（4）FROM /db2tbsp/backup1, /db2tbsp/backup2
```

参数说明如下。

☑ RESTORE DATABASE sample：表明要恢复的数据库映像。

☑ TABLESPACE(mytblspace1)：表明这是针对表空间的 RESTORE，并指定要恢复的表空间的名称。

☑ ONLINE：表明这是在线恢复。对于用户表空间，在线恢复和离线恢复都是允许的。

☑ FROM /db2tbsp/backup1, /db2tbsp/backup2：指定输入备份文件所在的位置。

2．关于表空间恢复的考虑

表空间被恢复以后，总是被置于 rollforward pending 状态。为了使表空间可以被访问并重置这个状态，必须至少将表空间前滚一个最小的时间点（minimum recovery time），这个最小的时间点可以确保表空间和日志与系统编目表中的信息一致。

考虑下面的这个例子：

（1）假设在时间 t1 上做了完整数据库备份，其中包括表空间 mytbls1。

（2）在时间 t2 上，在表空间 mytbls1 中创建了表 myTable。这将为表空间 mytbls1 到 t2 的恢复设置最小时间点。

（3）在时间 t3 上，使用 t1 时做的完整数据库备份恢复且只恢复表空间 mytbls1。

（4）恢复完成后，表空间 mytbls1 将被置于 rollforward pending 状态。如果可以前滚到最小时间点之前的一个时间点上，那么表空间 mytbls1 中将不包括表 myTable；然而，系统编目却说这个表的确在 mytbls1 中。为了避免出现这样的不一致，当恢复表空间时，DB2 将强制使你至少前滚到最小时间点上。

当对表空间或表空间中的表运行 DDL 语句时，最小时间点将被更新。为了确定表空间恢复的最小时间点，可以使用下面的命令：

```
list tablespaces show detail;
```

运行结果如图 15.8 所示。

3．使用控制中心执行恢复

除了可以使用命令行恢复数据库和表空间外，我们还可以使用控制中心进行恢复。图 15.9 展示了如何从控制中心调用 RESTORE 实用程序。要执行数据库或表空间恢复，可以在要恢复的数据库上单击鼠标右键并在弹出的快捷菜单中选择“复原”命令。

然后在弹出的“复原数据向导”对话框中选择要复原的映像，如图 15.10 所示。

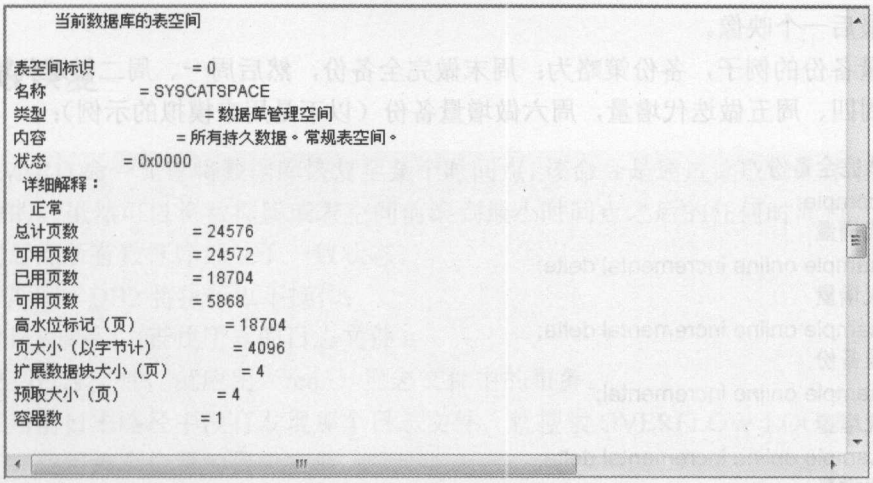


图 15.8　查看最小时间点

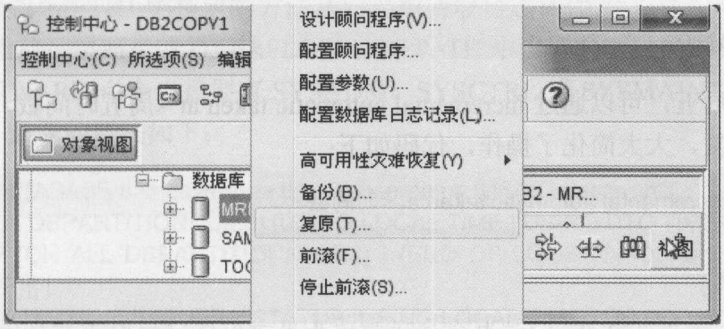


图 15.9　从控制中心调用 RESTORE 实用程序

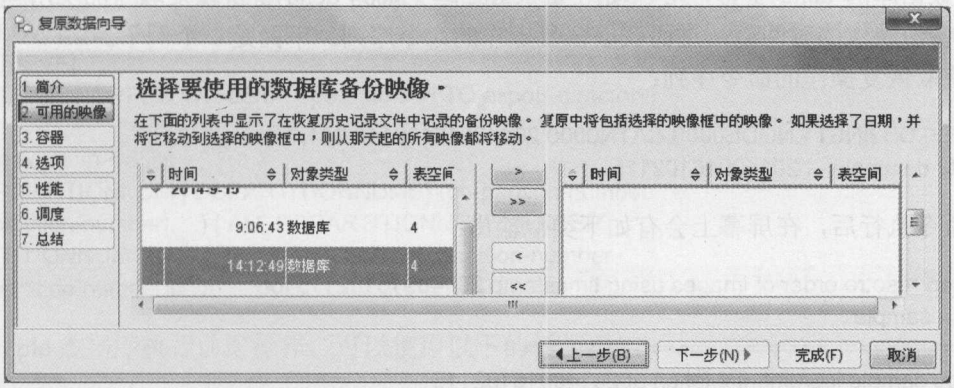


图 15.10　执行 RESTORE 实用程序所需的一些选项

4. 增量恢复

增量备份是只包含自从执行前一个备份以来更新过的页的备份映像。要使用增量备份来执行恢复，可以在 RESTORE DATBASE 命令上指定 INCREMENTAL 选项。通常，增量恢复会涉及一系列的恢复操作。

恢复中涉及的每个备份映像必须按以下顺序进行恢复：最近一次、第一次、第二次、第三次，依

此类推，直到最后一个映像。

以下是增量备份的例子，备份策略为：周末做完全备份，然后周一、周二做迭代增量备份，周三做增量备份，周四、周五做迭代增量，周六做增量备份（以下是用来模拟的示例）：

```
--周日做离线完全备份
backup db sample;
--周一做迭代增量
backup db sample online incremental delte;
--周二做迭代增量
backup db sample online incremental delta;
--周三做增量备份
backup db sample online incremental;
--周四做迭代增量
backup db sample online incremental delta;
--周五做迭代增量
backup db sample online incremental delta;
--周六做增量备份
backup db sample online incremental;
```

如果需要恢复到周五，可以通过 incremental automatic taken at <周五时间戳>语句，automatic 选项自动按照顺序进行恢复，大大简化了操作，代码如下：

```
restore db sample incremental automatic taken at <时间戳>;
```

5. 增量恢复检查

对于 DB2 数据库来说，其非 delta 备份的恢复操作是可以通过仅发出两条恢复命令来完成的。但如果要恢复数据库的 delta 备份，可以使用 db2ckrst 这一检测 delta 备份恢复顺序的实用程序，检测要恢复的增量备份映像的时间戳，进而获取恢复操作的命令序列。下面是一个例子。

（1）获取恢复操作的命令序列：

```
SAMPLE.0.DB2INST1.NODE0000.CATN0000.20140916102115.001;
db2ckrst –d sample –t 20140916102115
```

（2）命令执行后，在屏幕上会有如下类似输出：

```
Suggested resotre order of images using timestamp 20140916102115 for
Database sample .
==================================
restore db sample incremental taken at 20140916102115
restore db sample incremental taken at 20140916102325
restore db sample incremental taken at 20140916102115
==================================
```

（3）恢复 delta 备份：

```
restore db sample incremental taken at 20140916102115
restore db sample incremental taken at 20140916102325
restore db sample incremental taken at 20140916102115
```

15.5.2　前滚恢复

ROLLFORWARD 命令允许将数据库恢复至某个时间点，该命令是通过读取日志中相应的数据记录来完成这一过程的。虽然可以将数据库或表空间前滚到最小时间点之后的任何时间点，但是不能保证前滚到那个时间后点所有数据库都处于一致状态。

在前滚处理期间，DB2 将执行以下操作：

- ☑ 在当前日志路径中查找所需的日志文件。
- ☑ 如果找到日志文件，就应用（redo）日志文件中的事务。
- ☑ 如果在当前日志路径中没有发现那个日志文件，就搜索 OVERFLOW LOG PATH 选项指定的路径，并使用那个位置的日志文件。
- ☑ 如果在当前路径中没有发现那个日志文件，并且没有使用 OVERFLOW LOG PATH 选项，就使用 LOGARCHMETH1 中指定的方法从归档中获取日志文件。
- ☑ 如果日志在当前日志路径或 OVERFLOW LOG PATH 中，就重新应用事务。

要执行 ROLLFORWARD 命令，需要有 SYSADM、SYSCTRL 或 SYSMAINT 权限。

ROLLFORWARD 命令的语法如下：

```
ROLLFORWARD DATABASE database-alias [USER username [USING password]]
[TO {isotime [ON ALL DBPARTITIONNUMS] [USING LOCAL TIME | USING UTC TIME]} |
{END OF BACKUP [ON ALL DBPARTITIONNUMS]} | {END OF LOGS [On-DbPartitionNum-Clause]} [AND
{COMPLETE | STOP}]] |
[COMPLETE | STOP | CANCEL | {QUERY STATUS [USING LOCAL TIME | USING UTC TIME]}
[ON-DBPARTITIONNUM-CLAUSE]]
[TABLESPACE ONLINE | TABLESPACE (tblspace-name [{,tblspace-name}…])[ONLINE]]
[OVERFLOW LOG PATH (log-directory [{,log-directory ON DBPARTITIONNUM db-partition-number}…])]
[NORETRIEVE]
[RECOVER   DROPPED TABLE dropped-table-id TO export-directory]

on-dbpartitionnum-clause:
ON{{DBPARTITIONNUM | DBPARTITIONNUMS} (db-parition-number
[TO db-partition-number],…) | ALL DBPARTITIONNUMS [EXCEPT
{DBPARTITIONNUM | DBPARTITIONNUMS} (db-partition-number
[TO db-partition-number],…)]}
```

要对 sample 数据库执行前滚操作，可以使用以下的任何语句：

```
（1）ROLLFORWARD DATABASE sample TO END OF LOGS AND COMPLETE
（2）ROLLFORWARD DATABASE sample TO timestamp AND COMPLETE
（3）ROLLFORWARD DATABASE sample TO timestamp USING LOCAL TIME AND COMPLETE
```

从上面的代码中可以看出：

（1）在这个例子中，我们将前滚到日志的最后，这意味着所有归档的日志和活动日志都将被遍历。最后，sample 数据库将完成前滚，并通过回滚任何未提交的事务来脱离 rollforward-pending 状态。

（2）对于这个例子，DB2 将前滚到指定的时间点。在此语法中，时间必须是 UTC 时间。

（3）这个例子的时间戳可以用当地时间表示。

语法中没有关键字 OFFLINE，因为这是默认模式。对于 ROLLFORWARD 命令，这是可用于数据库前滚的唯一模式。

接下来看一下表空间的前滚。

表空间前滚通常可以在线进行或离线进行。一个例外就是系统编目表空间（SYSCATSPACE），这种表空间只能离线前滚。

下面是表空间前滚的示例：

```
ROLLFORWARD DATABASE sample TO END OF LOGS AND COMPLETE
TABLESPACE (userspace1) ONLINE
```

1. 关于表空间前滚的考虑

如果启用注册变量 DB2_COLLECT_TS_REC_INFO，那么只需要处理恢复表空间所需的日志文件。ROLLFORWARD 命令将忽略不需要的日志文件，这样可以缩短恢复时间。

ROLLFORWARD 命令的 QUERY STATUS 选项可用于列出：

☑ DB2 已经前滚的日志文件。

☑ 需要的下一个归档日志文件。

☑ 自前滚处理开始以来最近提交的事务的时间戳。

例如：

```
ROLLFORWARD DATABASE sample QUERY STATUS USING LOCAL TIME;
```

在某个表空间时间点前滚操作完成之后，表空间将被置于 backup-pending 状态。这里强制使用表空间或数据库的备份。

2. 使用控制中心执行前滚操作

除了可以使用命令行前滚数据库以外，还可以通过控制中心执行前滚操作。

实现步骤如下：

（1）在“控制中心”对象视图中，单击“所有系统”然后选择 MRKJ 数据库，然后在 MRKJ 数据库上单击鼠标右键，在弹出的快捷菜单中选择“前滚”命令，如图 15.11 所示。

（2）在“前滚向导”对话框中选择“前滚至某个时间点”和“归档日志的备用位置”，单击“完成”按钮。

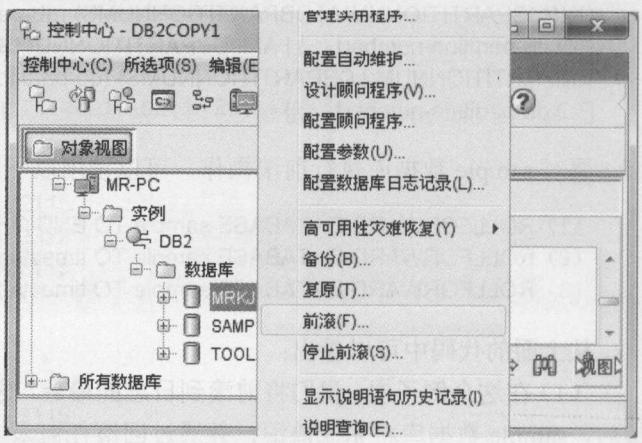


图 15.11 从控制中心调用 ROLLFORWARD 命令

15.5.3 崩溃恢复

对数据库执行的事务可能会被意外中

断。如果在发生故障时存在已开始但未提交的事务，或者存在已提交但未写入数据库中的事务，那么数据库就会处于不一致和不可用的状态。崩溃恢复的主要工作是将数据库恢复成一致可用的状态。为此，需要回滚没有提交的事务，并重做当发生崩溃时仍在内存中的已提交事务。

可以想象出来，如果数据库不具有崩溃恢复这一功能，当数据库发生宕机时，将使整个数据库报废或丢失宕机发生时的数据，而且数据库异常宕机的情况时有发生。

如果希望崩溃恢复的回滚是由数据库管理器自动完成的，那么应将数据库配置参数 autorestart 设置为 ON（默认值）以启动该自动重新启动参数。如果不想重新启动行为，可以将 austorestart 数据库配置参数设置为 OFF，如果这样的话，就需要在执行崩溃恢复时主动发出 RESTART DATABASE 命令。

需要注意的是，由于数据库在宕机后会处于不一致的状态，整个数据库都是不可用的，而恢复这一状态的唯一方式就是执行崩溃恢复。换句话说，崩溃恢复是恢复数据库为可用状态的必需环节。因此，建议将 autorestart 参数的值设为默认值 ON。当设置成 ON 后，在第一次连接数据库时，数据库就会自动启动崩溃恢复处理，直到崩溃恢复完成，数据库才可以使用。

15.6　常见的恢复场景

理解故障出现的原因以及如何从故障中恢复是非常重要的。下面将模拟各种不同的故障类型，并介绍从故障中恢复系统的方法。

15.6.1　整个数据库意外删除和损坏

【例 15.5】　如果数据库完全损坏，应如何恢复？（**实例位置：光盘\TM\sl\15\3**）

说明

完全数据库故障情况中，可以先通过应用已有的完全数据库备份文件恢复数据库，然后用日志文件通过前滚操作恢复到数据库损坏前的状态。这里使用 MRKJ 数据库做范例。

实现步骤如下。

（1）想要执行完全脱机数据库备份，应先用命令明确终止命令行处理器的后端进程。

```
TERMINATE;
```

如果应用程序连接到一个数据库，或者一个进程还在执行一个工作单元，TERMINATE 会导致丢失数据库连接。

（2）运行下面的命令使数据库处于一致状态：

```
FORCE APPLICATION ALL;
ROLLFORWARD DB mrkj TO END OF LOGS AND COMPLETE ;
--前滚至日志末尾，将应用所有剩余的日志文件，从而将数据库恢复到一致状态
```

如果提示数据库 mrkj 不处于前滚暂挂状态（已应用所有日志文件），则忽略此错误。

或者在“控制中心”对象视图中选择“所有系统”→“系统名”→“实例”→“实例名”→“数据库”→MRKJ，在“MRKJ”上单击鼠标右键，在弹出的快捷菜单中选择“前滚”命令。如果提示“此时不允许前滚操作，因为选择的数据库未处于前滚暂挂状态”，则单击“关闭”按钮直接进行第（3）步。在弹出的“前滚向导”对话框中选择“前滚至日志末尾”和“仅适用默认日志位置”，单击“完成”按钮。

（3）执行完全数据库脱机备份：

```
BACKUP DATABASE mrkj TO d:\mrkj\database;
```

注意

此处完全备份的数据库目录 d:\mrkj\database 必须已经存在。

或者在“控制中心”对象视图中选择“所有系统”→“系统名”→“实例”→“实例名”→“数据库”→MRKJ，在 MRKJ 上单击鼠标右键，在弹出的快捷菜单中选择“备份”命令。在弹出的“备份向导”对话框中单击“下一步”按钮，从而打开如图 15.12 所示的“备份向导”对话框。

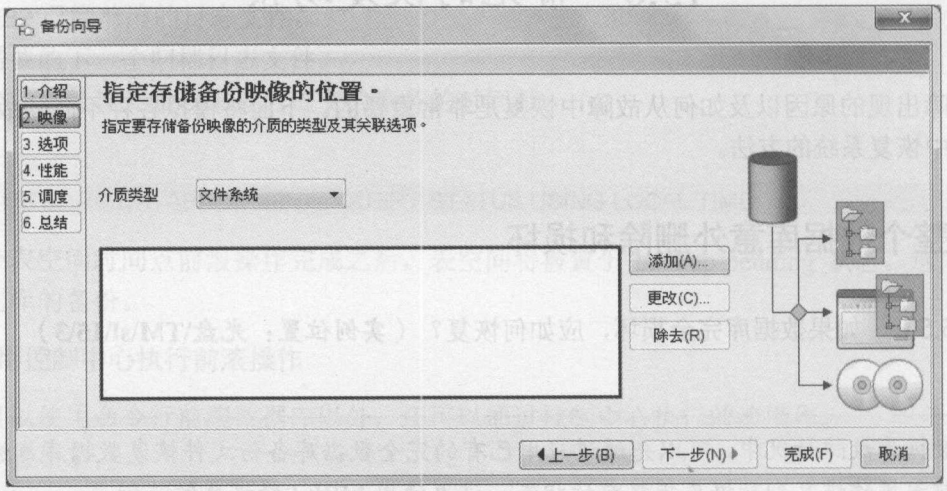


图 15.12 “备份向导”对话框

单击“添加”按钮可以在弹出的对话框中选择某个目录（例如，D:\mrkj\database，该目录必须已经存在）作为备份文件的位置，然后单击“确定”按钮，这时该文件夹就添加到了“介质类型”下面的列表中，单击“完成”按钮完成备份，如图 15.13 所示。

需要注意的是备份文件名称 ID，它类似于：20140917143926。这个 ID 只是备份映像的时间戳记，恢复过程中需要使用它。

（4）要模拟故障场景，应先明确终止命令行处理器的后端进程：

```
TERMINATE;
```

再完全删除数据库：

```
FORCE APPLICATION ALL;
DROP DATABASE mrkj;
```

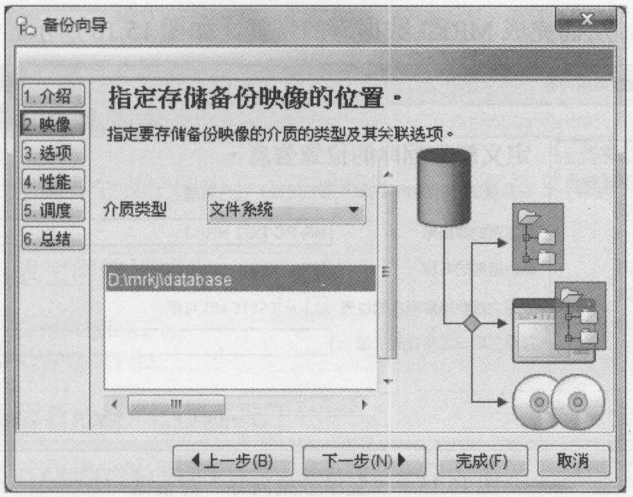


图 15.13　“备份向导”对话框

（5）从备份映像恢复数据库。

本例中使用时间戳记为 20140917143926 的备份映像恢复 MRKJ 数据库，运行下面的命令：

```
RESTORE DATABASE mrkj FROM d:\mrkj\database TAKEN AT 20140917143926 TO "D: " INTO MRKJ;
```

或者在“控制中心”对象视图中选择“所有系统”→“系统名”→“实例”→“实例名”→“数据库”，在“数据库”文件夹上单击鼠标右键，在弹出的快捷菜单中选择“创建数据库”→“从备份”命令，如图 15.14 所示。

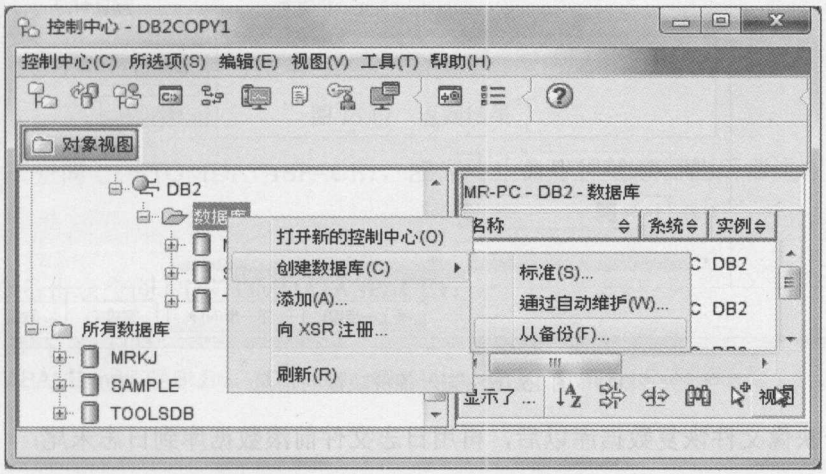


图 15.14　打开“从备份”对话框

在弹出的“复原数据向导”对话框中输入要复原的数据库名称“MRKJ”，输入新的数据库名称“MRKJ”，输入复原之后数据库所在的位置“D:”，单击“下一步”按钮，如图 15.15 所示。

在“复原数据向导”对话框中选中“手工输入备份映像信息”单选按钮，然后在“介质类型”下拉列表框中选择“文件系统”，单击“添加”按钮，添加备份文件的位置（这里是 D:\mrkj\database），输入日期“2014 年 9 月 17 日”，时间“14:39:26”（日期和时间由第（3）步完成后生成的备份映像确

定），单击“完成”按钮，从而完成 MRKJ 数据库的恢复，如图 15.16 所示。

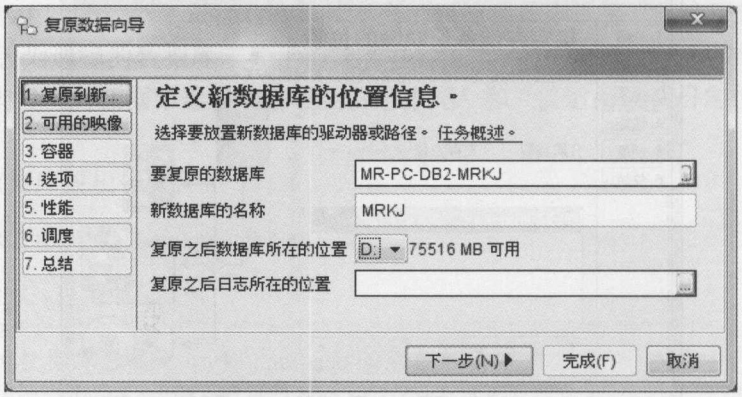


图 15.15 “复原数据向导”对话框

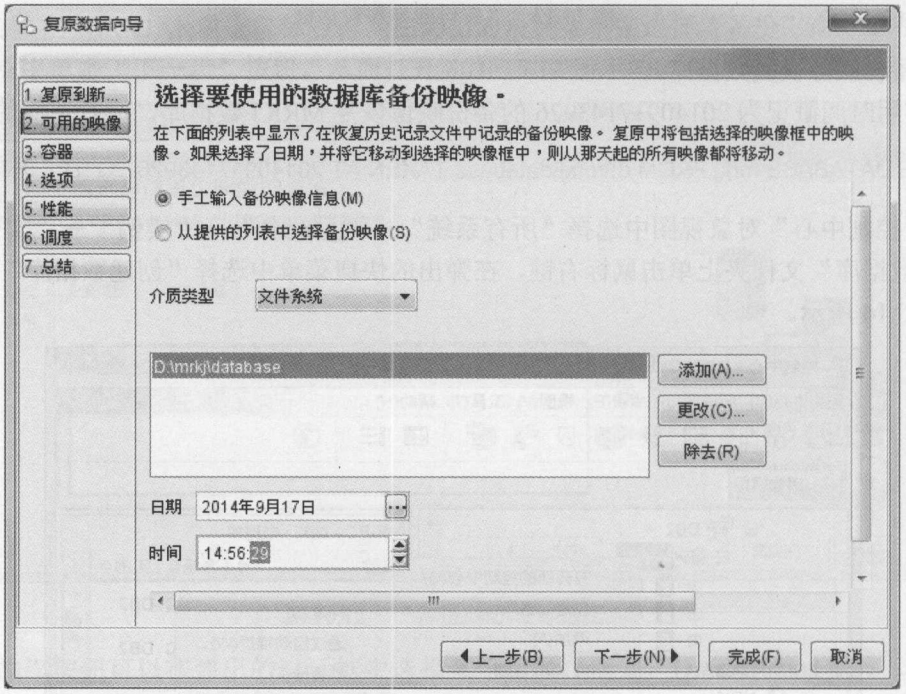


图 15.16 输入备份映像的信息

（6）用备份映像文件恢复数据库以后，再用日志文件前滚数据库到日志末尾：

```
ROLLFORWARD DB mrkj TO END OF LOGS AND COMPLETE;
```

（7）验证数据库及其中的对象仍然与备份前一致：

```
CONNECT TO mrkj USER mr USING 1;
LIST TABLESPACES SHOW DETAIL;
LIST TABLES;
```

上面几条命令返回所有表空间及表中的数据集合，应该与执行备份时的状态一致。

15.6.2　表空间容器意外丢弃或损坏

【例 15.6】　如果一个或多个表空间容器丢失或损坏，应如何恢复数据库系统？

实现步骤如下。

（1）获取定义的所有表空间的目录：

```
CONNECT TO mrkj USER mr USING 1;
LIST TABLESPACES SHOW DETAIL;
```

（2）获取所有表空间容器信息的目录：

```
LIST TABLESPACE CONTAINERS FOR 3 SHOW DETAIL;
```

上面命令中的“3”是本环境中表空间 CUSTOMDATASPACE1 的表空间 ID（从本书的前几章中可看出，本书大多数数据库对象示例都在 CUSTOMDATASPACE1 表空间上，如果这个表空间损坏，数据丢失会比较严重）。它是通过之前 LIST TABLESPACES SHOW DETAIL 命令中的输出获得的。所使用的每个表空间 ID 都需要重复这个命令。返回结果如图 15.17 所示。

表空间 3 的表空间容器

容器标识	= 0
名称	= E:\DB2\NODE0000\MRKJ\T0000003\C0000000.LRG
类型	= 文件
总计页数	= 8192
可用页数	= 8188
可访问	= 是

图 15.17　返回结果

（3）要备份表空间 CUSTOMDATASPACE1，应先终止命令行处理器的后端进程：

```
TERMINATE;
```

再用以下命令备份表空间 CUSTOMDATASPACE1：

```
FORCE APPLICATION ALL;
BACKUP DATABASE mrkj TABLESPACE customdataspace1 TO d:\mrkj\tablespaces;
```

注意

此处备份表空间容器的目录 d:\mrkj\tablespaces 必须已经存在。

或者在“控制中心”对象视图中选择“所有系统”→“系统名”→“实例”→“实例名”→“数据库”→MRKJ，在 MRKJ 上单击鼠标右键，在弹出的快捷菜单中选择“备份”命令，在弹出的“备份向导”对话框中选择“映像”，在“指定存储备份映像位置”对话框中，“介质类型”下拉列表框选择“文件系统”，单击“添加”按钮，添加备份文件的位置（例如，d:\mrkj\tablespaces，该目录必须存在），单击“完成”按钮，从而完成备份。

也需要注意备份文件的时间戳记，它类似于：20140917143926。

说明

在一个或多个表空间容器发生故障的情况中，可以先通过应用已有的表空间容器备份文件恢复表空间容器，然后用日志文件通过前滚操作恢复到表空间容器损坏前的状态。这里使用 MRKJ 数据库作为示范。

（4）模拟表空间故障。

手动模拟此场景，即表空间容器文件被某个用户意外删除。

关闭控制中心和命令编辑器，手动找到表空间 CUSTOMDATASPACE1 所在的文件夹 E:\DB2\NODE0000\MRKJ\T0000003\C0000000.LRG，并将其删除。其中文件名和路径是由第（2）步中的命令返回的结果决定的。

随后当连接到数据库并尝试执行与表空间 CUSTOMDATASPACE1 相关的操作时，将返回错误。例如：

```
connect to mrkj user mr using 1;
create table mr.table_test(columnl INT) IN customdataspace1;
```

返回以下错误消息：

```
SQL0290N  不允许访问表空间  SQLSTATE=55039
```

还可以使用以下命令检查表空间状态：

```
LIST TABLESPACES SHOW DETAIL
```

删除容器后，上面的命令将显示 TS1 的状态为 0x400，该代码表示脱机且不可访问状态。由于删除了 C0000000.LRG 表空间容器文件，因此表空间不再处于正常状态（0x000）。

再次执行 LIST TABLESPACE CONTAINES 命令，可以验证该容器不可访问：

```
LIST TABLESPACE CONTAINERS FOR 3 SHOW DETAIL;
```

显示结果“可访问=否”。

（5）为了恢复表空间备份映像，先终止命令行处理器后端进程：

```
TERMINATE;
```

迫使本地或远程用户或应用程序断开 DB2 系统，以便维护 DB2 服务器：

```
FORCE APPLICATION ALL;
```

再使用以下命令恢复表空间备份映像：

```
RESTORE DATABASE mrkj TABLESPACE (customdataspace1) FROM d:\mrkj\tablespacees
TAKEN AT 20140917143926;
```

这时表空间容器文件 E:\DB2\NODE0000\MRKJ\T0000003\C0000000.LRG 又出现了。

或者在“控制中心”对象视图中选择“所有系统”→“系统名”→“实例”→“实例名”→“数据库”→MRKJ 文件夹，在 MRKJ 文件夹上单击鼠标右键，在弹出的快捷菜单中选择“复原”命令，从而打开“复原数据向导”对话框，选中“复原到现有数据库”单选按钮，如图 15.18 所示。

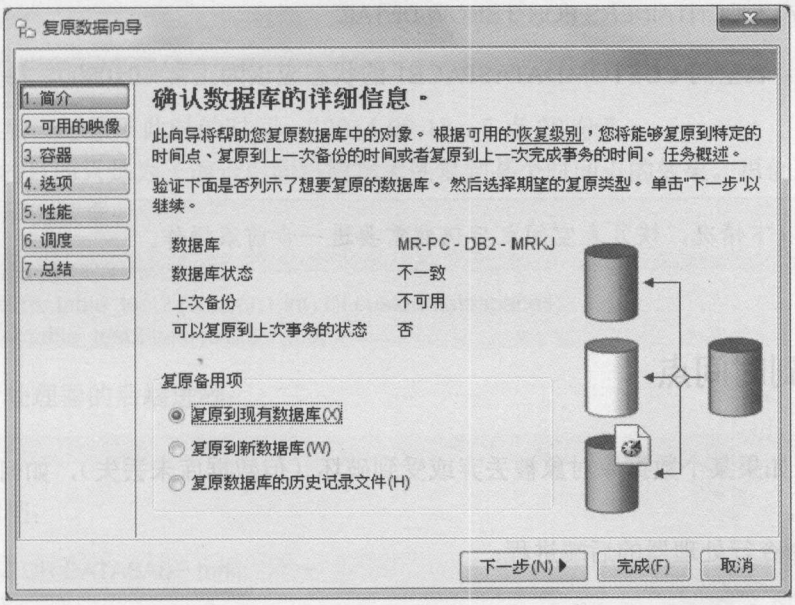


图 15.18　选择“复原备用项”

单击“下一步”按钮。在“选择要使用的数据库备份映像”界面中选中“从提供的列表中选择备份映像”单选按钮，再选择“2014-9-17 14:39:26 表空间”（日期和时间由第（3）步完成后生成的备份映像确定），然后单击“>”按钮，这时这个表空间映像就到了“选择的备份映像”栏中，如图 15.19 所示，单击“完成”按钮，从而完成 CUSTOMDATASPACE1 表空间的恢复。

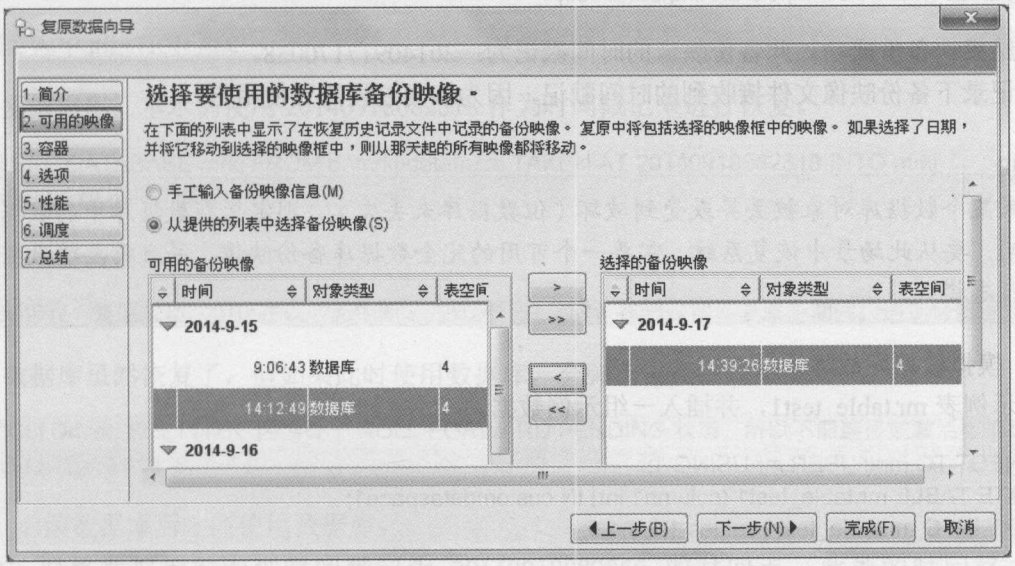


图 15.19　选择要使用的数据库备份映像恢复表空间

（6）检查表空间状态，确保容器可以访问：

```
CONNECT TO mrkj USER mr USING 1;
LIST TABLESPACES SHOW DETAIL;
LIST TABLESPACE CONTAINERS FOR 3 SHOW DETAIL;
```

如果恢复成功，表空间 CUSTOMDATASPACE1 的状态应该为正常（0x000），并且所有容器应该都可以访问。



注意

可能会遇到以下情况，恢复表空间之后仍然需要进一步前滚操作。

15.6.3 恢复到时间点

【例 15.7】 如果某个数据库对象被丢弃或受到破坏（但数据库未丢失），如何恢复到时间点？实现步骤如下。

（1）先终止命令行处理器的后端进程：

```
TERMINATE;
```

迫使本地或远程用户或应用程序断开 DB2 系统，以便维护 DB2 服务器：

```
FORCE APPLICATION ALL;
```

再执行完全数据库备份：

```
BACKUP DATABASE mrkj TO d:\mrkj\database;
```

执行结果：备份成功。此备份映像的时间戳记为：20140917170058。

务必记录下备份映像文件接收到的时间戳记，因为恢复过程需要使用它。



说明

如果某个数据库对象被丢弃或受到破坏（但数据库未丢失），则定义在数据库中的数据库对象不可访问。要从此场景中恢复系统，需要一个可用的完全数据库备份映像，并且需要将数据库配置为归档日志模式。

（2）模拟数据库故障场景。

创建示例表 mr.table_test1，并插入一组示例数据。

```
CONNECT TO mrkj USER mr USING 1;
CREATE TABLE mr.table_test1 (column1 int) IN customdataspace1;
INSERT INTO mr.table_test1 values 1,2,3,4,5;
```

为了让此场景更接近现实，使用一些命令强制 DB2 打开归档文件。

先终止命令行处理器的后端进程：

```
TERMINATE;
```

再将日志归档：

```
ARCHIVER LOG FOR DATABASE mrkj;
```

上述命令完成后，记录下此时的时间：“2014-09-18　8 点 28 分”。

这里记录时间的目的只是为了演示后面将数据库恢复到某个时间点的效果。再创建示例表 mr.table_test2，并插入一组示例数据。

```
CONNECT TO mrkj USER mr USING 1;
CREATE TABLE mr.table_test2 (column1 int) IN customdataspace1;
INSERT INTO mr.table_test2 values 5,6,7,8,9;
```

先终止命令行处理器的后端进程：

```
TERMINATE;
```

然后将日志归档：

```
ARCHIVE LOG FOR DATABASE mrkj;
```

执行完成后记录此时的时间：“2014-09-18　8 点 32 分”。

假设此时数据库出现故障，需要恢复数据库。

（3）先终止命令行处理器的后端进程：

```
TERMINATE;
```

迫使本地用户、远程用户或应用程序断开 DB2 系统，以便维护 DB2 服务器：

```
FORCE APPLICATION ALL;
```

再恢复数据库，本示例使用 20140918082318 作为时间戳记来进行恢复。

```
RESTORE DATABASE mrkj FROM d:\mrkj\database TAKEN AT 20140918082318 INTO mrkj
WITHOUT PROMPTING;
```

系统提示警告信息：

```
SQL2540W　复原成功，但是在以“无中断”方式进行处理时，在“数据库复原”期间，遇到警告“2539”。
```

此时数据库虽然恢复了，但如果此时使用数据库，会弹出错误：

```
[IBM][CLI Driver] SQL1117N 因为处于 ROLL-FORWARD PENDING 状态，所以不能连接或激活数据库"MRKJ"。
 SQLSTATE=57019
```

（4）前滚数据库后才能使用数据库。

例如，要将数据库前滚到时间戳记为 20140918082850 的时间点，此处的时间格式必须是

ROLLFORWARD 工具可识别的形式，即 2014-09-18.8.28.50。

```
ROLLFORWARD DATABASE mrkj TO 2014-09-18.8.28.50 USING LOCAL TIME AND STOP;
```

DB2 将应用指定时间戳之前的所有日志。

除了使用命令方式，也可以在“控制中心”对象视图中选择“所有系统”→“系统名”→“实例”→“实例名”→“数据库”→MRKJ 文件夹，在 MRKJ 文件夹上单击鼠标右键，在弹出的快捷菜单中选择“前滚”命令。在弹出的“前滚向导”对话框中选择“前滚至某个时间点—本地时间”选项，输入向前滚到的时间点（例如，2014-09-18.8.28.50），单击“下一步”按钮。在下一个对话框中，选择“完成前滚并返回到活动状态”，单击“完成”按钮，从而完成前滚至时间点的操作。

此时可以打开 MRKJ 数据库，并看到出现表 mr.table_test1。前面创建的表 mr.table_test2 并没有恢复，因为这个表是 2014-09-18.8.28.50 以后出现的。

15.7 小　　结

本章主要对 DB2 中数据的备份和恢复进行了讲解，并通过具体的实例说明了备份和恢复的使用方法。通过本章的学习，读者应该能够掌握常用的 DB2 备份和恢复功能及它们的使用方法，并能够在实际应用中使用这些功能。

15.8 实践与练习

（1）恢复操作有哪些种类？（**答案位置：光盘\TM\sl\15\4**）

（2）什么是事务日志记录？（**答案位置：光盘\TM\sl\15\5**）

（3）离线备份数据库和在线备份数据库分别使用什么命令？（**答案位置：光盘\TM\sl\15\6**）

第16章

PureXML 技术应用

（视频讲解：55 分钟）

IBM 的新一代数据库第一次实现了关系型引擎与层次型引擎的结合——PureXML 技术。在 PureXML 中，XML 作为一种新的数据类型，允许 XML 格式的文件存储到 DB2 中 XML 数据类型的列中。本章将要揭开 PureXML 技术的神秘面纱。

通过阅读本章，您可以：

- 理解 PureXML 技术
- 掌握创建可以存储 XML 数据的 DB2 数据库
- 掌握创建具有 XML 列的表
- 了解样本数据
- 掌握 XML 模式存储库
- 理解将 XML 文档插入到 XML 类型的列中

16.1 PureXML 技术简介

关系型数据库很早就已经开始考虑对 XML 的支持，但传统的技术一般还是基于关系型数据库的基本框架，用表之间的关系去模拟 XML 的层次结构，关系数据库中的第一代 XML 支持切分（或分解）文档以适应关系表格或将文档原封不动地存储为字符或二进制大对象（CLOB、BLOB 或 DBCLOB）。这两种方法中的每一种都尝试将 XML 模型强制转换成关系模型。

IBM 的新一代数据库 DB2 V9 以上版本第一次实现了关系型引擎与层次型引擎的结合，实现了混合数据库，IBM 将此技术称为“PureXML 技术”。

与过去关系型数据库的 XML 增强功能不同，PureXML 技术及混合型数据库的出现，超越了关系型数据库固有的局限性，为数据库的应用及开发开辟了新的领域。PureXML 技术其意义不仅仅是提供了一种存储 XML 数据的更有效的方式，而是其还可以用于使用 XML 数据的各类领域。同时，由于 PureXML 所代表的层次型数据组织方式，使得我们有了关系型和层次型两种数据建模的方法，即使是对于传统非 XML 的关系型数据，在某些情况下将传统关系型的数据转化为层次型存储也将带来巨大的业务价值。

如图 16.1 所示，在两种数据存储（XML 和关系）的顶部是一个混合型数据库引擎，该引擎可以处理 XQuery、Xpath、SQL 和 SQL/XML。该引擎采用带有 SQL 和 XQuery 解析程序的双语查询编译器。为了查询 XML 数据，DB2 提供了两种语言：SQL/XML 和 XQuery。因此开发人员可以根据具体情况更适用的原则使用任何一种语言（或同时使用这两种语言）。使用混合型 DB2，用户可以根据信息管理的需要来灵活地选择数据的存储组织模式和访问方法，实现二者之间灵活的转换。

注意

XPath 是 XQuery 的子集，本书中所讲述的 XQuery 都包含 XPath。

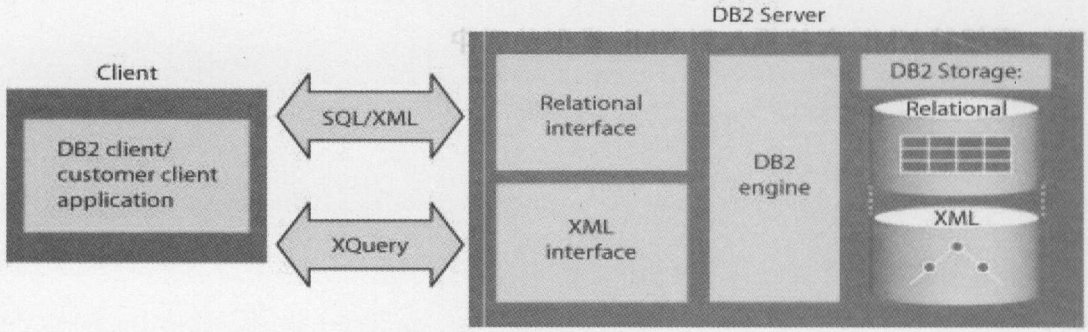


图 16.1 XML 与关系数据的集成

混合型系统并不要求所有的数据都表示成关系数据，也不要求所有数据都使用 XML 格式，相反，设计者应该为适当的任务选择适当的模型。

DB2 还提供了一个 XML 模式库（XSR）用来注册并保持 XML 模式，并使用这些模式验证 XML 文件。每个 XML 文件基础的模式验证是可选的，因此，模式和文件之间的联系是由该文件决定的，而不是由该列决定的，这提供了极大的灵活性。

16.2　创建可以存储 XML 数据的 DB2 数据库

必须在使用 UTF-8 代码集合兼容的地域代码定义的数据库中创建具有 XML 列的表。

下面的命令可创建用于存储 XML 数据的数据库，该数据库使用 UTF-8 代码集：

```
CREATE DATABASE database_name USING CODESET UTF-8 TERRITORY CN;
```

参数说明如下。

- ☑ TERRITORY CN：表示采用“中华人民共和国”地域代码。
- ☑ 默认情况下，将在创建数据库时启用自动存储器。自动存储器可以提高性能并使得管理 XML 数据更容易，因为它生成可以根据需要扩展的数据库管理的空间（DMS）表空间。

注意

（1）在创建包含 XML 数据的数据库时，代码集必须选择 UTF-8，且要选择兼容的地域代码。如果创建数据库时选择其他字符集，则接下来在该数据库中建表后，如果表中有 XML 类型的字段且该字段为中文名称时，那么往表中插入数据时会报错且无法插入 XML 字段，原因是如果数据库的代码集不是 UTF-8，则插入 XML 数据时自动将 XML 文档的字符集转换为 UTF-16，这与数据库代码集冲突。

（2）如果选择将 XML 数据存储在使用 UTF-8 以外的代码集的数据库中，最好以不进行代码页转换的方式插入此数据。用户可打开“计算机”→“属性”→“高级系统设置”，在系统属性中选择“高级”→然后选择“环境变量”→“系统变量”，新建 ENABLE_XMLCHAR 系统变量，其值为 NO，以阻止 XML 解析期间使用字符数据类型，从而阻止可能的字符替换。

下面介绍创建具有 XML 数据的表空间、创建具有 XML 列的表、将 XML 列添加到现有表、将其他数据类型的列修改成 XML 列、将 XML 列修改成其他类型的列、删除表中的 XML 列等操作，这些操作与数据库中普通数据类型的操作类似，只不过是把相关列都变成了 XML 列而已。

16.3　创建具有 XML 列的表

通过 DB2 界面创建具有 XML 列的表与通过 DB2 界面创建普通表的方式类似，只需要在向表中添

加列时定义一个或多个列是 XML 数据类型即可。或者在通过 CREATE TABLE 语句创建表时定义一个或多个列为 XML 数据类型。表中每一行的 XML 数据类型的列可以存储一个 XML 格式的文件。即使每个 XML 文件逻辑上与表中的一行关联，XML 列的关系列存储方式也不同。整型、字符型等常用的其他数据类型的列将数据直接存储在列中。而 XML 列仅包含列的描述符，数据单独存储。

定义 XML 列、插入或查询 XML 数据时并不需要 XML 模式。XML 列可以存储没有模式的文档，正如许多经过演化后变得不同的 XML 模式文档。

定义 XML 列时不要指定长度。但是，与 DB2 数据库交换的已序列化 XML 数据的大小限制为每个 XML 类型的值为 2GB，因此，XML 文档的有效限制为 2GB。

为了讨论 XML 数据类型在 DB2 中的应用，先在 MRKJ 数据库中创建一个包含 XML 列的示例表“联系方式表”。

【例 16.1】 在 MRKJ 数据库中创建包含 XML 列的表“联系方式表”，表结构如表 16.1 所示（**实例位置：光盘\TM\sl\16\1**）。

表 16.1 “联系方式表”结构

列 名	数据类型	长 度	可 空	默认值	说 明
学号	CHARACTER	6	否	无	主键
联系信息	XML		是	无	无

实现步骤如下：

（1）连接到 MRKJ 数据库：

```
CONNECTION mrkj USER mr USING 1;
```

（2）创建“联系方式表”：

```
CREATE TABLE mr.联系方式表
(
  学号 CHARACTER(6) NOT NULL PRIMARY KEY,
  联系信息 XML
)IN customdataspace1;
```

说明

指定主键是可选的，并不是存储 XML 或对 XML 建立索引必需的。

16.4 样本数据

“联系方式表”中的样本数据如表 16.2 所示。

表 16.2　“联系方式表”的样本数据

学　号	联 系 信 息
14001	<联系方式 xmlns=http://mr.com 姓名="马芳"> <email>MF@106.com</email> <电话>176××××××××</电话> <地址 国家="中国"> <邮政编码>510000</邮政编码> <省或直辖市>广东省</省或直辖市> <市或县>广州</市或县> <详细地址>南沙区银华街 48 号</详细地址> </地址> </联系方式>
14002	<联系方式 xmlns=http://mr.com 姓名="宋静"> <email>SJ@106.com</email> <电话>136××××××××</电话> <地址 国家="中国"> <邮政编码>150711</邮政编码> <省或直辖市>黑龙江省</省或直辖市> <市或县>哈尔滨</市或县> <详细地址>高尔基路 3 号</详细地址> </地址> </联系方式>
14003	<联系方式 xmlns=http://mr.com 姓名="周天星"> <email>ZTX@106.com</email> <电话>176××××××××</电话> <地址 国家="中国"> <邮政编码>102511</邮政编码> <省或直辖市>吉林省</省或直辖市> <市或县>梅河口</市或县> <详细地址>保利路 3 号</详细地址> </地址> </联系方式>
14004	<联系方式 xmlns=http://mr.com 姓名="蒋阳泉"> <email>JYQ@163.com</email> <电话>158××××××××</电话> <地址 国家="中国"> <邮政编码>130052</邮政编码> <省或直辖市>吉林省</省或直辖市> <市或县>吉林</市或县> <详细地址>丰满区</详细地址> </地址> </联系方式>

续表

学　号	联 系 信 息
14005	<联系方式 xmlns=http://mr.com 姓名="董敏"> <email>DM@106.com</email> <电话>158××××××××</电话> <地址 国家="中国"> <邮政编码>120011</邮政编码> <省或直辖市>广东省</省或直辖市> <市或县>深圳</市或县> <详细地址>福田区石厦路 3567 号</详细地址> </地址> </联系方式>
14006	<联系方式 xmlns=http://mr.com 姓名="万方"> <email>WF@106.com</email> <电话>189××××××××</电话> <地址 国家="中国"> <邮政编码>1563251</邮政编码> <省或直辖市>吉林省</省或直辖市> <市或县>长春</市或县> <详细地址>南关区上海路 18 号</详细地址> </地址> </联系方式>
14007	<联系方式 xmlns=http://mr.com 姓名="秦东雪"> <email>QDX@106.com</email> <电话>189××××××××</电话> <地址 国家="中国"> <邮政编码>320056</邮政编码> <省或直辖市>湖北省</省或直辖市> <市或县>武汉</市或县> <详细地址>武昌区公正路 2288 号</详细地址> </地址> </联系方式>
14008	<联系方式 xmlns=http://mr.com 姓名="欧阳云"> <email>Y@int.com</email> <电话>132××××××××</电话> <地址 国家="中国"> <邮政编码>210011</邮政编码> <省或直辖市>江苏省</省或直辖市> <市或县>南京</市或县> <详细地址>鼓楼区菜市路 1088 号</详细地址> </地址> </联系方式>

续表

学　号	联 系 信 息
14009	<联系方式 xmlns=http://mr.com 姓名="董淑芬"> <email>SF@inti.com</email> <电话>139××××××××</电话> <地址 国家="中国"> <邮政编码>685023</邮政编码> <省或直辖市>福建省</省或直辖市> <市或县>厦门</市或县> <详细地址>龙口区盎然路 69 号</详细地址> </地址> </联系方式>
14010	<联系方式 xmlns=http://mr.com 姓名="蔡传奇"> <email>CQ@106.com</email> <电话>186××××××××</电话> <地址 国家="中国"> <邮政编码>130223</邮政编码> <省或直辖市>辽宁省</省或直辖市> <市或县>大连</市或县> <详细地址>西岗区金石路 1123 号</详细地址> </地址> </联系方式>
14969	<联系方式 xmlns=http://mr.com 姓名="赵亮"> <email>ZL@ttwo.com</email> <电话>177××××××××</电话> <地址 国家="中国"> <邮政编码>130011</邮政编码> <省或直辖市>辽宁省</省或直辖市> <市或县>大连</市或县> <详细地址>中山区老汇路 1128 号</详细地址> </地址> </联系方式>

说明

（1）“xmlns=http://mr.com”用于设定默认的 XML 名称空间，名称空间 http://mr.com 是在 XML 模式存储库（XSR）中注册的某个 XML 模式中定义的，该 XML 模式用于验证输入到“联系信息”列的 XML 文档是否符合该 XML 模式规范。也就是说“联系方式”“姓名”“email”“电话”“地址”等都在名称空间 http://mr.com 中。例如，“联系方式”完整的限定名称（即“Qname”，后面将讲解限定名称）为“xmlns: 联系方式”；“电话”完整的限定名称为“xmlns: 电话”；“地址”完整的限定名称为“xmlns: 地址”。

（2）学号为“14001”～“14010”和“14969”的学生的“联系信息”列都符合 contact_schema 模式（本书后面将介绍该 XML 模式）。

16.5 XML 模式存储库

在关系数据库中存储 XML 数据会提供可靠性、可管理性以及 RDBMS 的优点。XML 已经成为 Web 上数据表示集成和交换的标准，随着 Web 服务和面向服务结构的出现，企业与企业之间，企业内部之间的信息交换通常都借助于 XML。

16.5.1 XML 模式存储库概念

XML 模式存储库（XML Schema Repository，XSR）支持创建一份信息，作为 XSR 对象包含在 XML 模式、DTD 或外部实体中。在向 XML 列插入数据或更新 XML 数据时，此信息用来验证和处理 XML 数据。使用得最多的 XSR 对象是 XML 模式。XSR 中的每个 XML 模式都可以由一个或多个 XML 模式文档组成。在 XML 模式由多个文档组成的情况下，用于开始注册过程的文档是“XML 模式主文档”。在 XML 模式只由一个文档组成的情况下，该文档就是 XML 模式主文档。

16.5.2 增大应用程序堆栈大小配置参数

在注册大多数 XML 模式时，需要增大应用程序堆大小（APPLHEAPSZ）配置参数。

【例 16.2】 增大 MRKJ 数据库的应用程序堆大小配置参数。

实现步骤如下：

（1）在“控制中心”的对象视图中选择 MRKJ 数据库，在其右键快捷菜单中选择“配置参数”命令，从而打开“数据库配置”对话框，如图 16.2 所示。

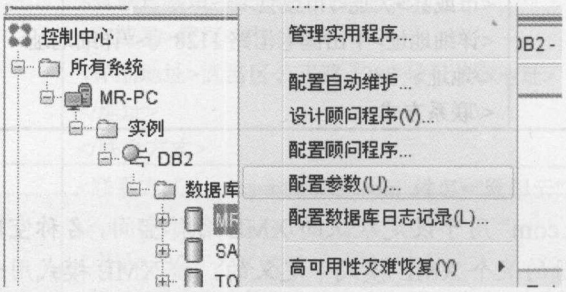


图 16.2 配置数据库参数菜单

（2）在“数据库配置”对话框中找到“性能”目录下的“APPLHEAPSZ”参数行，如图 16.3 所示。单击 APPLHEAPSZ 后面的文本框的“…”按钮，从而打开“更改数据库配置参数”对话框，如图 16.4 所示。

（3）在“更改数据库配置参数”对话框中修改应用程序堆大小的值为“1024”，注册 XML 模式时

推荐 1024 页以上的应用程序堆。如果操作 XML 数据时发生溢出，可以参照上面的方法继续增大应用程序堆的大小，如图 16.5 所示。

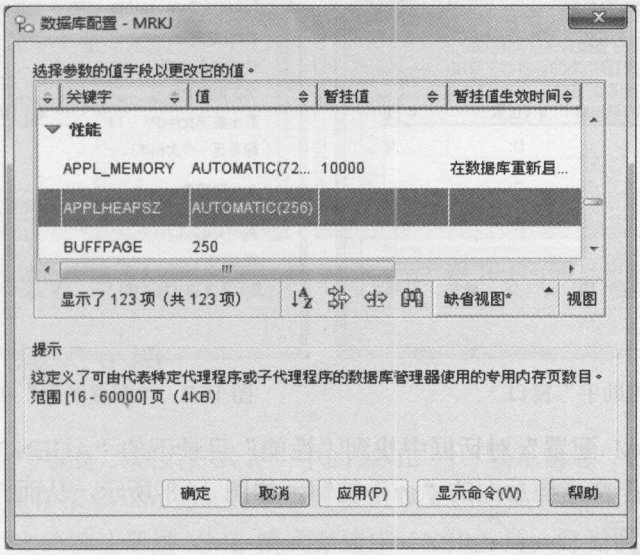


图 16.3　数据库配置

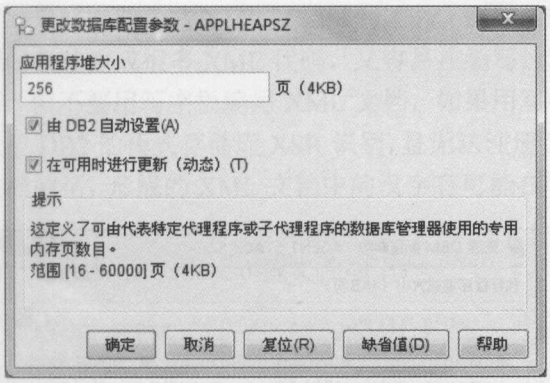


图 16.4　“更改数据库配置参数”对话框

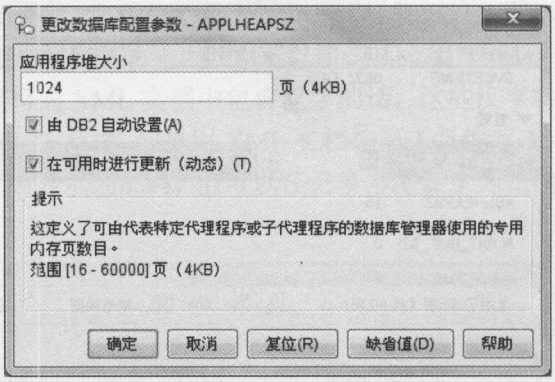


图 16.5　更改数据库配置参数

（4）在所有对话框中都单击“确定”按钮，完成修改，在可用时修改生效。

16.5.3　增大代理程序堆栈大小配置参数

要在 Windows 32 位操作系统上注册非常复杂的 XML 模式，还需要增大代理程序堆栈大小（agent_stack_sz）配置参数。

【例 16.3】　增大 MRKJ 数据库的代理程序堆栈大小配置参数。

实现步骤如下：

（1）选择“开始”→“所有程序”→IBM DB2→“DB2COPY1（默认值）”→“设置工具”→“配置助手”命令，弹出“配置助手”窗口，如图 16.6 所示。在“配置助手”窗口的主菜单中选择“配置”

菜单项，再单击“DBM 配置”，如图 16.7 所示。

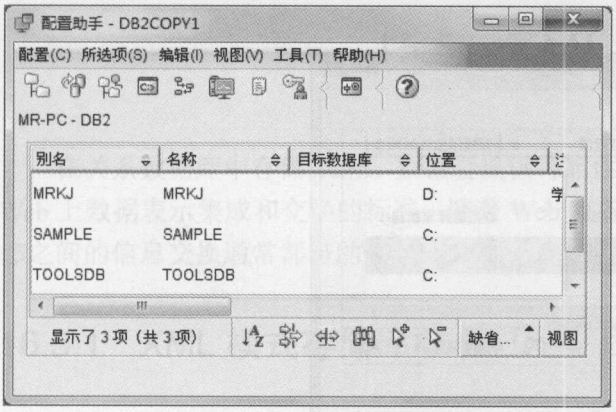


图 16.6 “配置助手”窗口

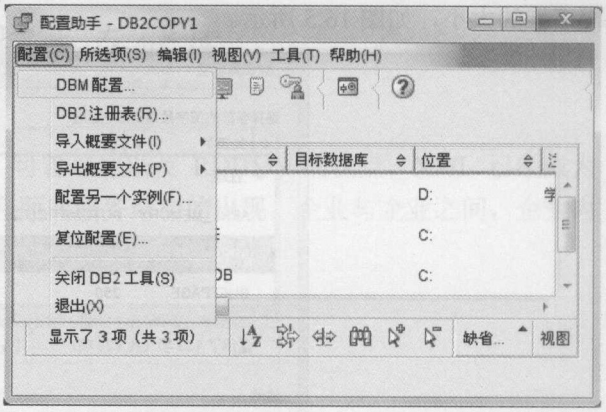


图 16.7 “配置助手”中的“DBM 配置”菜单

（2）在弹出的“DBM 配置”对话框中找到“性能”目录下的“AGENT_STACK_SZ”参数，单击“AGENT_STACK_SZ”文本框后面的“…”按钮，如图 16.8 所示，从而打开“更改 DBM 配置参数”对话框，如图 16.9 所示。

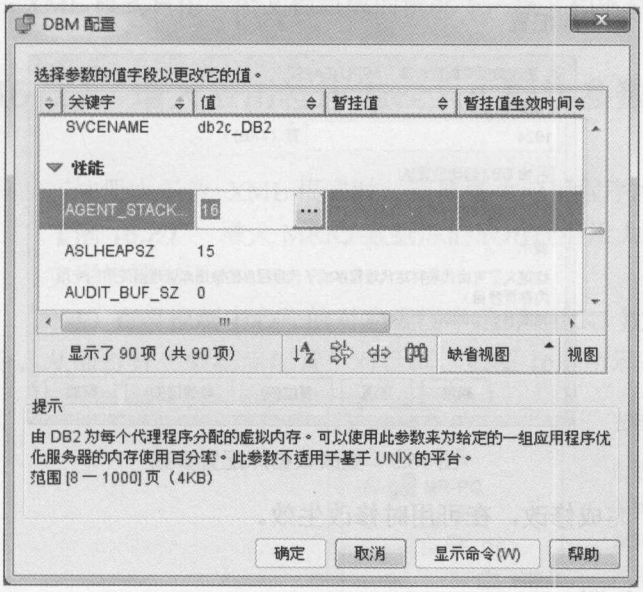


图 16.8 AGENT_STACK_SZ 参数

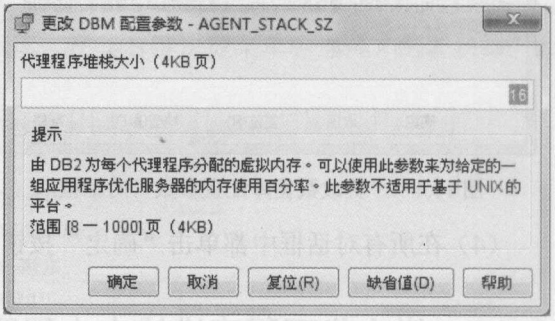


图 16.9 更改 DBM 配置参数

（3）在“更改 DBM 配置参数”对话框中，修改“代理程序堆栈大小“为“256”，如图 16.10 所示。注册 XML 模式时推荐 256 页以上的代理程序堆栈。

（4）在弹出的“DB2 消息”对话框中单击“关闭”按钮，然后单击“确定”按钮，完成修改，如图 16.11 所示。

重新启动 DB2 后修改生效。如果操作 XML 数据时发生溢出，可以参照上面的方法继续增大代理程序堆栈的大小。

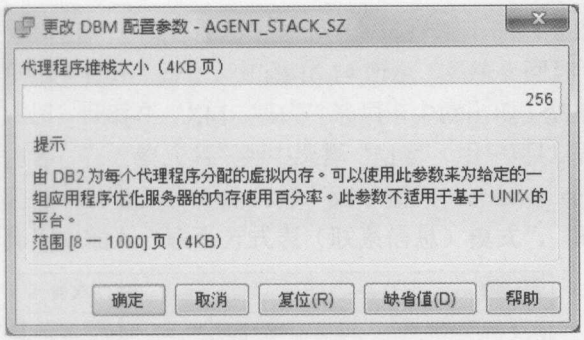


图 16.10　更改代理程序堆栈大小

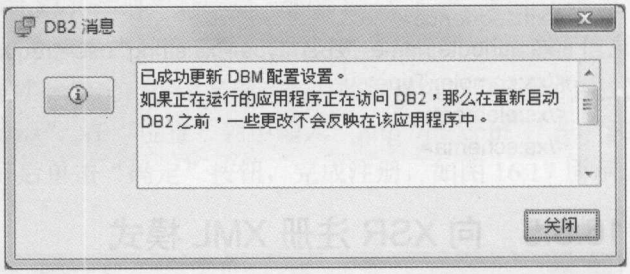


图 16.11　成功更新了 DBM 配置参数

16.5.4　编写 XML 模式文档

【例 16.4】　编写用于验证“联系方式表”中的 XML 列“联系信息”的 XML 模式文档 contact_schema.xsd，并将该文档保存在 D:\xmldata 目录下（**实例位置：光盘\TM\sl\16\2**）。

文档的用途：当这个文档被注册进 XML 模式存储库（XSR）后，每当向“联系方式表”中“联系信息”XML 列插入数据时，用户可以选择验证输入的 XML 文档是否符合这个 XML 模式。

最好借助一些 XML 文档编写工具，例如 Altova Xmlspy 或 IBM Data Studio，这样既可以在图形界面中自动生成很多 XML 代码，又容易在编写过程中检查文档中的错误，也可以用记事本编写 XML 文档，但不要用写字板编写 XML 文档，如果用写字板编写的 XML 文档中包含汉字的话，这些汉字将不能被 DB2 的模式存储库 XSR 识别，且无法注册 XML 模式；反之，如果用 XML 文档编写工具编写 XML 文档的话，形成的 XML 文档中的汉字有可能在写字板中显示为乱码，但可以在记事本中显示为正确的汉字。

contact_schema.xsd 源文件如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="http://mr.com"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
<xs:element name="联系方式">
<xs:complexType>
      <xs:sequence>
      <xs:element name="email" type="xs:string" minOccurs="1" maxOccurs="unbounded"/>
      <xs:element name="电话" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="地址" minOccurs="0" maxOccurs="unbounded">
            <xs:complexType>
<xs:sequence>
            <xs:element name="邮政编码" type="xs:string" minOccurs="0"/>
            <xs:element name="省或直辖市" type="xs:string" minOccurs="0"/>
            <xs:element name="市或县" type="xs:string" minOccurs="0"/>
            <xs:element name="详细地址" type="xs:string" minOccurs="0"/>
            </xs:sequence>
                  <xs:attribute name="国家" type="xs:string"/>
```

```
</xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute name="姓名" type="xs:string" use="required"/>
</xs:complexType>
</xs:element>
</xs:schema>
```

16.5.5 向 XSR 注册 XML 模式

XML 模式定义了 XML 文件的结构，包括元素定义、属性定义、命名空间、代码页等。在向数据库插入 XML 值之前，这些模式可以用来验证 XML 值是否符合 XML 模式要求。这些模式通常存储在文件系统中，利用它们进行验证之前需要将它们注册到数据库。注册模式到数据库使得这个模式成为数据库的一个实体，不需要再依赖操作系统去阅读模式文档的内容。

在将 XML 模式、DTD 或外部实体用于验证和注释之前，它必须先进行 XML 模式存储库（XSR）注册。

可以用下面任何一种方法将 XML 模式注册到 XSR：

☑ DB2 图形界面（通过“控制中心”将 XML 模式注册到 XSR）。

☑ 命令（通过“命令编辑器”或“命令行处理器”将 XML 模式注册到 XSR）。

☑ 调用存储过程（从应用程序向 XSR 注册 XML 模式）。

注意

通过 DB2 图形界面将 XML 模式注册到 XSR 的过程实质上就是调用命令将 XML 模式注册到 XSR。也就是说，前两种方法是等价的。

1. 通过“控制中心”将 XML 模式注册到 XSR

【例 16.5】 通过“控制中心”将例 16.4 中创建的用于验证“联系方式表”中 XML 列“联系信息”的 XML 模式文档“D:\xmldata\contact_schema.xsd”注册到 XSR 中，形成名称为“mr.contact_schema”的 XML 模式，其中 mr 是 XML 工件的模式，将 mr.contact_schema 模式绑定到 URI：“http://mr.com”。

实现步骤如下：

（1）在“控制中心”对象视图中选择 MRKJ 数据库下“XML 模式存储库（XSR）”文件夹，在该文件夹上单击鼠标右键，在弹出的快捷菜单中选择“向 XSR 注册”命令，如图 16.12 所示。

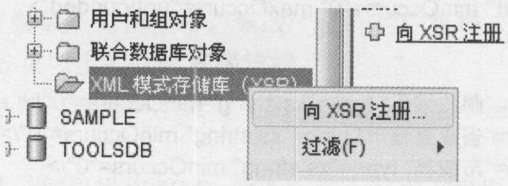


图 16.12 控制中心向 XSR 注册 XML 模式菜单

（2）在弹出的“向 XSR 注册”对话框中，单击“文件名”后面的“…”按钮，如图 16.13 所示，

会弹出“选择文件名”对话框，选择例 16.4 中创建的 XML 模式文档“D:\xmldata\contact_schema.xsd”，单击“打开”按钮，如图 16.14 所示，系统会回到“向 XSR 注册”对话框，如图 16.15 所示。单击“添加”按钮，此时在“XML 模式”选项卡中就出现了注册这个模式的一行信息，如图 16.16 所示。修改这行信息内容：在“模式名”列中选择“MR”（用户可以根据实际情况自行定义模式名，此处也可以输入新的模式名），在“XML 工件名称”列中输入“contact_schema”，在“位置”列中输入“http://mr.com”，在“注释”中输入“联系方式表（联系信息）模式”，填好后单击“确定”按钮，完成注册，如图 16.17 所示。

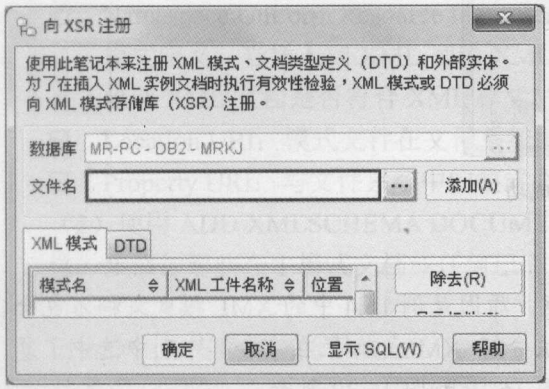


图 16.13　“向 XSR 注册”对话框

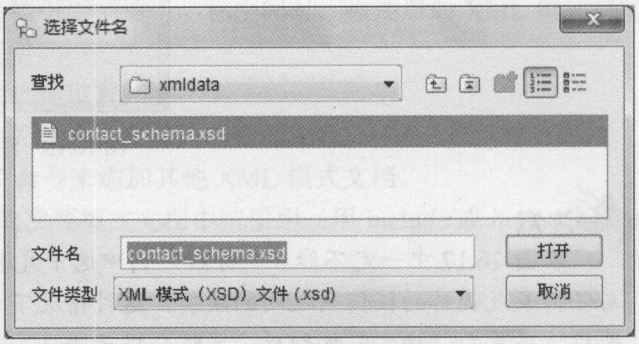


图 16.14　“选择文件名”对话框

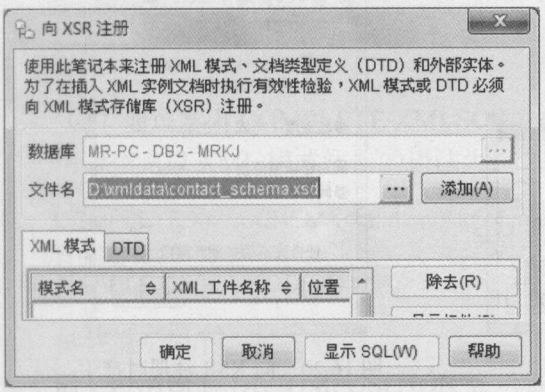


图 16.15　向 XSR 注册

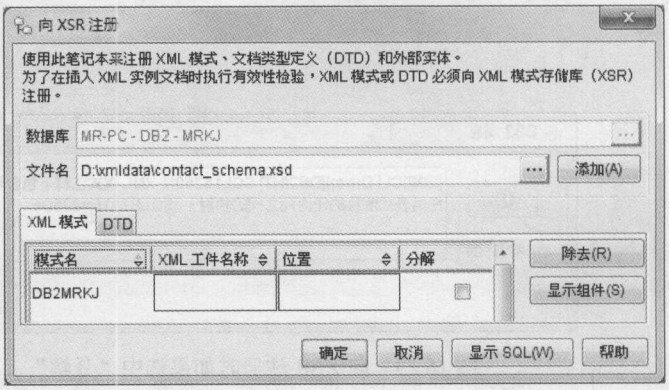


图 16.16　添加 contact_schema.xsd

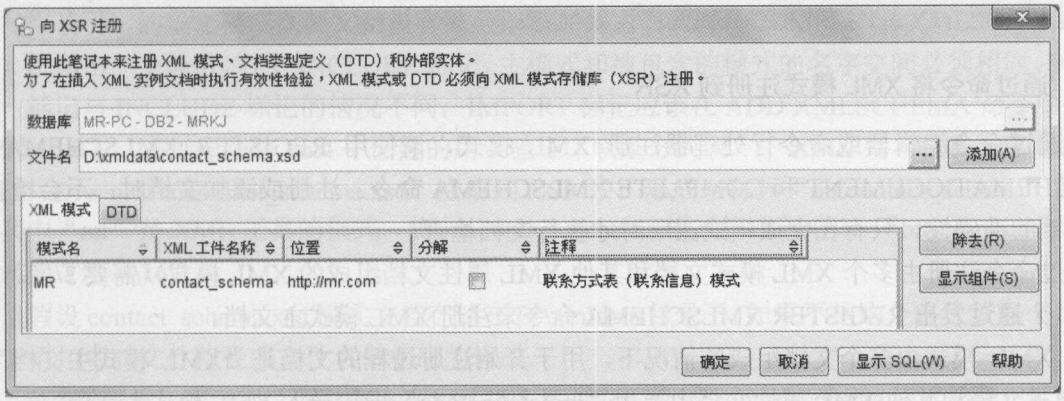


图 16.17　设置注册信息

注册成功后会出现图 16.18 所示的对话框，单击“关闭”按钮即可完成整个注册过程。

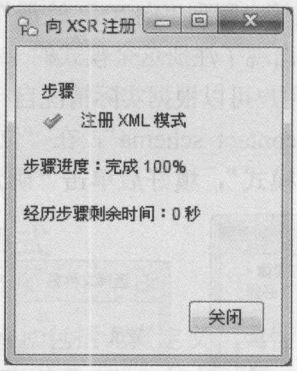


图 16.18　注册成功

注意

在图 16.17 中一定不能在“分解”列中选中复选框，原因是例 16.4 中的 XML 模式文档不包含与任何表或昵称的任何列之间的映射，这种情况下无法分解 XML。如果在“分解”列中选中了复选框，当单击“确定”按钮后，系统会提示错误，如图 16.19 和图 16.20 所示。

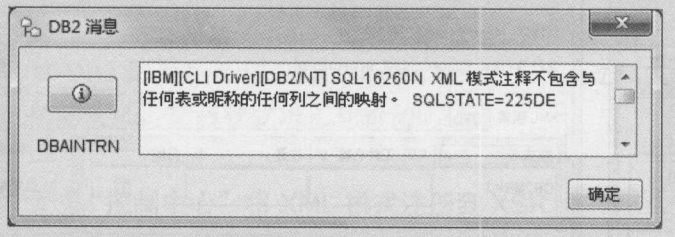


图 16.19　向 XSR 注册时如果选中“分解”复选框则产生错误

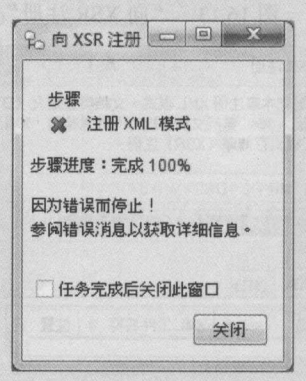


图 16.20　向 XSR 注册时产生错误从而注册失败

2．通过命令将 XML 模式注册到 XSR

要通过命令编辑器或命令行处理器注册 XML 模式，请使用 REGISTER XMLSCHEMA、ADD XMLSCHEMA DOCUMENT 和 COMPLETE XMLSCHEMA 命令。注册或添加文档时，不会检查 XML 模式文档的准确性。只有在完成模式注册时才执行文档检查。

通过命令注册由多个 XML 模式文档和其他 XML 属性文档组成的 XML 模式，需要 3 步。

（1）通过发出 REGISTER XMLSCHEMA 命令来注册 XML 模式主文档。

在 XML 模式由多个文档组成的情况下，用于开始注册过程的文档是“XML 模式主文档”。其他 XML 模式文档和其他 XML 属性文档中要用“import/include”语句导入 XML 模式主文档。

用命令注册主模式文档的语法为：

```
REGISTER XMLSCHEMA <namespace URI>
FROM <location URI>
WITH <property URI>
AS <relational id>
```

参数说明如下。

☑ Namespace Uniform Resource Identifier（URI）：XML 文档的命名空间，这个命名空间由 XML 模式定义，当插入到 XML 列的 XML 文档引用了这个名称空间时，被注册的 XML 模式就能验证 XML 文档是否符合 XML 模式的规定。

☑ Location URI：模式文件在文件系统中的物理位置。

☑ Property URI：与文件系统中的模式文件关联的属性文件的物理位置。

（2）使用 ADD XMLSCHEMA DOCUMENT 命令来添加其他 XML 模式文档。

这一步添加那些在主模式文档或任何已添加的次要模式文档中被引用（用 include 或 import 标记）的 XML 模式文档。

用命令添加模式文档的语法为：

```
ADD XMLSCHEMA DOCUMENT TO<relation id of main schema>
ADD <document URI>
FROM <location URI>
WITH <property URI>
```

（3）通过发出 COMPLETE XMLSCHEMA 命令来完成注册。

用命令完成 XML 模式注册的语法为：

```
COMPLETE XMLSCHEMA<relational id of main schema>
WITH <property URI>
[ENABLE DECOMPOSITION];
```

参数说明如下。

☑ ENABLE DECOMPOSITION 表示将 XML 文档分解为关系表，如果不需要将 XML 文档分解为关系表，则不要用这个子句。

☑ 当被引用的主文档或次要文档没有被全部添加到主模式时，COMPLETE XMLSCHEMA 命令将抛出一个异常。对于 INCLUDE 标记，主模式和被包含的模式的名字空间必须相同。IMPORT 标记与 INCLUDE 标记的情况不同，IMPORT 标记应该在 ADD XMLSCHEMA 命令中提到。

【例 16.6】 通过命令将 16.4 中创建的用于验证“联系方式表”中 XML 列“联系信息”的 XML 模式文档“D:\xmldata\contact_schema.xsd”注册到 XSR 中，形成名称为“mr.contact_schema”的 XML 模式，其中“mr”是 XML 工具的模式，将 mr.contact_schema 模式绑定到 URI：“http://mr.com”（**实例位置：光盘\TM\sl\16\3**）。

这里假设 contact_schema.xsd 模式文档存储在 D:\xmldata 目录下，如果不是这个路径或这个文件名，请在以下语句中对文件路径或文件名做相应的修改。

在“命令编辑器”中运行以下命令等效于例 16.5 界面中注册 mr.contact_schema 模式的操作过程。

实现步骤如下。

（1）连接 MRKJ 数据库：

```
CONNECT TO MRKJ USER MR USING 1;
```

（2）注册 mr.contact_schema 模式：

```
REGISTER XMLSCHEMA 'http://mr.com'
    --目标名称空间
FROM D:\xmldata\contact_schema.xsd
    --XML 模式文档位置
AS mr.contact_schema
    --注册的 XML 模式名
COMPLETE;
    --已完成注册，不需要添加其他模式文档和模式属性文档
```

（3）给新注册的 mr.contact_schema 模式添加注释：

```
COMMENT ON XSROBJECT mr.contact_schema IS '联系方式表（联系信息）模式';
```

因为此 XML 模式只由一个模式文档组成，所以可将注册和完成步骤组合成单个命令。

3．查看注册成功的 XML 模式

成功向 XSR 注册 XML 模式后可以通过在控制中心查看或通过 SQL 命令查看注册成功的 XML 模式。

【例 16.7】 在“控制中心”查看例 16.5 中注册的 XML 模式。

在“控制中心”对象视图中选择 MRKJ 数据库下“XML 模式存储库（XSR）”文件夹，在右面的内容窗格中可以看到已经创建“XML 工件名称”为“CONTACT_SCHEMA”的 XML 工件。说明成功注册了该 XML 模式，如图 16.21 所示。

MR-PC - DB2 - MRKJ - XML 模式存储库（XSR）

XML 工件名称	模式名	目标名称空间	类型	注释
CONTACT_SCHEMA	MR	http://mr.com	XML ...	联系方式表（联系信息）模式

图 16.21 查看已经注册的 XML 格式

通过查询 SYSCAT.XSROBJECTS 目录视图（它包含有关存储在 XSR 中的对象的信息），也可以验证是否成功注册了该 XML 模式。

【例 16.8】 用命令查看例 16.5 中创建的模式（**实例位置：光盘\TM\sl\16\4**）。

实现步骤如下。

（1）连接 MRKJ 数据库：

```
CONNECT TO MRKJ USER MR USING 1;
```

（2）通过查询 SYSCAT.XSROBJECTS 目录视图查询已经注册的 XML 模式：

```
select objectschema,objectname from syscat.xsrobjects;
```

查询结果如图 16.22 所示。

```
OBJECTSCHEMA                    OBJECTNAME
------------------------------ ------------------------------ ------------
MR                              CONTACT_SCHEMA
```

图 16.22　查询已经注册的 XML 模式

4．通过存储过程将 XML 模式注册到 XSR

在高级程序语言中不能使用“命令行处理器”或“命令编辑器”。创建数据库时，会自动创建用于注册 XML 模式的存储过程。如果要用高级程序编程语言注册 XML 模式，需要使用 CALL 语句来调用 XSR_REGISTER、XSR_ADDSCHEMADOC 和 XSR_COMPLETE 存储过程。注册或添加文档时，不会检查 XML 模式文档的准确性，只有在完成 XML 模式注册时才执行文档检查。

（1）SYSPROC.XSR_REGISTER

此存储过程用来注册 XML 模式：

```
XSR_REGISTER(rschema,name,schemalocation,content,docproperty);
```

参数说明如下。

- ☑ rschema：XML 模式的关系模式名。此参数的类型为 VARCHAR(128)。如果这个参数是 NULL，表示是“当前模式”。
- ☑ name：XML 模式的关系标识符。此参数的类型为 VARCHAR(128)。XML 模式的全名为 rschema.name。
- ☑ schemalocation：主 XML 模式的名称空间。此参数的类型为 VARCHAR(1000)。
- ☑ content：XML 模式文档的内容。此参数的类型为 BLOB（30MB）。这是一个非空参数。
- ☑ docproperty：模式关联的属性文档的内容。此参数的类型为 BLOB(5MB)。

（2）SYSPROC.XSR_ADDSCHEMADOC

调用这个存储过程将 XML 模式文档添加到主模式：

```
XSR_ADDSCHEMADOC(rschema,name,schemalocation,content,docproperty);
```

参数说明如下。

- ☑ rschema：XML 模式的关系模式名。此参数的类型为 VARCHAR(128)。如果这个参数是 NULL，表示是“当前模式”。
- ☑ name：XML 模式的关系标识符。此参数的类型为 VARCHAR(128)。XML 模式的全名为 rschema.name。
- ☑ schemalocation：主 XML 模式的名称空间。此参数的类型为 VARCHAR(1000)。
- ☑ content：XML 模式文档的内容。此参数的类型为 BLOB(30MB)。这是一个非空参数。
- ☑ docproperty：模式关联的属性文档的内容。此参数的类型为 BLOB(5MB)。

（3）SYSPROC.XSR_COMPLETE

用下面命令完成注册：

```
XSR_COMPLETE(rschema,name,schemaproperty,isUsedForDecomp);
```

参数说明如下。

- ☑ schemaproperty：此参数的类型为 BLOB(5MB)，如果 XML 模式有属性，则 schemaproperty 指

定属性。如果没有关联属性或表示 XML 模式属性的 XML 文档，则这个参数的值也可以是 NULL。

☑ isUsedForDecomp：INTEGER 类型。如果这个参数是 1，则表示将这个 XML 模式分解为关系模式；否则是 0，表示不分解。

【例 16.9】 假设 XML 模式由两个 XML 模式文档组成：PO.xsd 和 address.xsd，这两个文档都存储在本地“C:\TEMP”中；用户想要根据这两个 XML 模式文档用名称 user1.Poschema 注册 XML 模式。XML 模式有一个与它关联的属性文件，称为 schemaProp.xml，此属性文件也存储在同一个本地“C:\TEMP”目录中。用户定义了 URI，通过它，此模式在外部称“http://myPOschema/PO”（也就是 XML 名称空间）。如何在高级编程语言中注册由 XML 模式文档 PO.xsd 和 address.xsd 以及 XML 属性文件 schemaProp.xml 组成的 XML 模式？（**实例位置：光盘\TM\sl\16\5**）



说明

上述的 XML 模式文档 PO.xsd 和 address.xsd 以及属性文件 schemaProp.xml 只是为了描述在高级程序编程语言中注册 XSR 对象的语法而假设的上述情况，实际上本书并没有创建这些文档。

在高级编程语言中需要通过调用存储过程，注册由 XML 模式主文档和其他 XML 文档组成的 XML 模式，注册过程分为 3 步。

实现步骤如下。

（1）通过调用 SYSPROC.XSR_REGISTER 存储过程来注册 XML 模式主文档：

```
CALL SYSPROC.XSR_REGISTER
('user1','POschema','http://myPOschema/PO',:content_host_var,NULL)
```

（2）完成注册之前，添加要随主 XML 模式一起包括的任何其他 XML 模式文档。对于示例来说，此步骤不是可选的，因为 XML 模式由两个 XML 模式文档组成，这两个文档都必须已经注册。使用 XSR_ADDSCHEMADOC 存储过程来添加其他 XML 模式文档。在以下示例中，将 address 的模式构造添加至 XSR 对象：

```
CALL SYSPROC.XSR_ADDSCHEMADOC
('user1','POschema','http://myPOschema/address',:content_host_var,NULL)
```

（3）通过调用 SYSPROC.XSR_COMPLETE 存储过程来完成注册。在以下示例中，最后一个参数指示不会将 XML 模式用于分解（如果值为 1，那么指示它将用于分解）：

```
CALL SYSPROC.XSR_COMPLETE ('user1','POschema',:schemaproperty_host_var,0);
```

16.5.6 删除 XSR 中的 XML 模式

1. 用界面删除 XSR 模式

【例 16.10】 删除 XSR 中的 XML 模式 contact_schema。

实现步骤如下：

（1）在“控制中心”对象视图中选择 MRKJ 数据库中的“XML 模式存储库（XSR）”文件夹，在控制中心里面的内容窗格上选择名称为“CONTACT_SCHEMA”的 XML 文件，在其右键快捷菜单中选择“删除”命令，如图 16.23 所示。



图 16.23　删除 CONTACT_SCHEMA 选项

（2）在弹出的“确认”对话框中，单击“确定”按钮，完成删除，如图 16.24 所示。

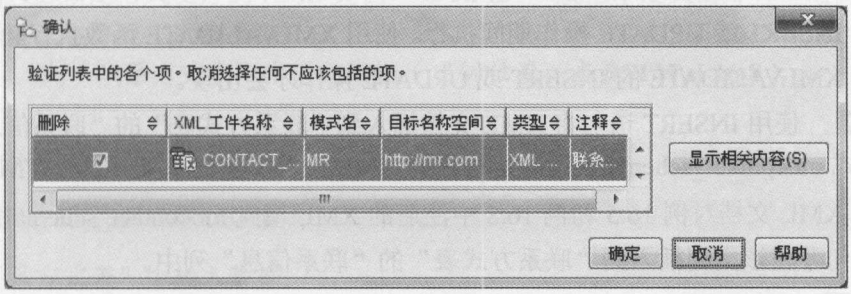


图 16.24　单击“确定”按钮完成删除

2．用命令删除 XSR 中的 XML 模式

在“命令编辑器”中运行下面的命令可删除 XSR 中的 XML 模式：

```
DROP XSROBJECT xsrobject-name;
```

【例 16.11】　用命令删除 mr.contact_schema 模式（**实例位置：光盘\TM\sl\16\6**）。

此方法等效于使用界面删除 XML 模式。

实现步骤如下。

（1）连接 MRKJ 数据库：

```
CONNECT TO MRKJ USER MR USING 1;
```

（2）删除 mr.contact_schema 模式：

```
DROP XSROBJECT mr.contact_schema;
```

3．调用 XSR_REMOVE 存储过程删除 XSR 模式

在高级程序编程语言中用 CALL 语句调用 XSR_REMOVE(rschema,name)存储过程，可以删除 XSR 中的 XML 模式。

16.6 将 XML 文档插入到 XML 类型的列中

想要在 XML 列中插入数据，可以使用 INSERT 或 UPDATE 语句，还可选择用 XML 模式验证即将插入的 XML 文档。

16.6.1 插入时针对 XML 模式验证 XML 文档

只能针对 XML 模式验证 XML 文档，不支持 DTD 验证（虽然不能针对 DTD 进行验证，但仍可以插入包含 DOCTYPE 或引用 DTD 的文档）。

验证通常在 INSERT 或 UPDATE 操作期间执行。使用 XMLVALIDATE 函数执行验证。只有当验证成功时，指定了 XMLVALIDATE 的 INSERT 或 UPDATE 操作才会出现。

【例 16.12】 使用 INSERT 语句将 XML 文档插入到“联系方式表”的“联系信息”列中，插入时针对 XML 模式 mr.contact_schema 验证即将插入的 XML 文档（**实例位置：光盘\TM\sl\16\7**）。

只有当新的 XML 文档对例 16.5 和例 16.8 中注册的 XML 模式 mr.contact_schema 来说是有效时，以下 INSERT 语句才将该文档插入到“联系方式表”的“联系信息”列中。

实现步骤如下。

（1）连接 MRKJ 数据库：

```
CONNECT TO MRKJ USER MR USING 1;
```

（2）用 INSERT 语句将一个 XML 文档插入到“联系方式表”的“联系信息”列中，插入时针对 XML 模式 mr.contact_schema 验证即将插入的 XML 文档：

```
INSERT INTO mr.联系方式表(学号,联系信息)
VALUES ('14001',XMLVALIDATE(XMLPARSE(DOCUMENT
'<联系方式 xmlns="http://mr.com" 姓名="马芳">
<email>MF@106.com</email>
<电话>176××××××××</电话>
<地址 国家="中国">
<邮政编码>510000</邮政编码>
<省或直辖市>广东省</省或直辖市>
<市或县>广州</市或县>
<详细地址>南沙区银华街 48 号</详细地址>
</地址>
</联系方式>'
PRESERVE WHITESPACE)
ACCORDING TO XMLSCHEMA ID mr.contact_schema));
```

说明

（1）可以用类似上面的命令向“联系方式表”中插入表 16.2 中的其他数据。

（2）XMLVALIDATE 函数对 XML 数据进行验证。因为此示例中的 XML 文档是作为字符数据传递的，所以 XMLVALIDATE 函数必须与 XMLPARSE 函数配合使用。XMLPARSE 函数将其自变量解析为 XML 文档并返回 XML 值。

（3）只能在 INSERT、UPDATE 或 DELETE 语句中将字符数据直接指定给 XML 列，此处使用 INSERT INTO 语句。

16.6.2　插入时不验证 XML 文档

XML 列的输入必须是格式良好的 XML 文档，如 XML1.0 规范中定义的那样。对应的应用程序数据类型可以是 XML、字符或二进制类型。

【例 16.13】　向“联系方式表”中插入数据（**实例位置：光盘\TM\sl\16\8**）。

实现步骤如下。

（1）连接 MRKJ 数据库：

```
CONNECT TO MRKJ USER MR USING 1;
```

（2）向“联系方式表”中插入数据：

```
insert into mr.联系方式表 (学号,联系信息)
values ('12100', '<contact 姓名="李娜">
<邮箱>sunny@126.com</邮箱>
</contact>')
```

说明

上面语句的功能是将“联系方式表”中学生的 XML 列“联系信息”的数据复制到“学生表”中的 XML 列“联系方式”中，两表的连接条件是“学号”相同。

16.7　小　　结

本章主要对 DB2 中的新技术——PureXML 技术进行详细讲解。在具体讲解过程中，首先介绍 PureXML 技术，让读者对什么是 PureXML 技术有一个清晰的了解；然后讲解了创建可以存储 XML 数据的 DB2 数据库、创建具有 XML 列的表、样本数据、XML 模式存储库、将 XML 文档插入到 XML 类型的列中等内容。通过本章的学习，读者应该熟练掌握 PureXML 技术的使用，并能够使用 PureXML 技术解决数据库开发中遇到的问题。

16.8 实践与练习

（1）PureXML 与以前处理数据库中 XML 数据的方法相比有什么优势？（**答案位置：光盘\TM\sl\16\9**）

（2）在创建包含 XML 数据的数据库时，如何避免 XML 数据中的中文显示为乱码？（**答案位置：光盘\TM\sl\16\10**）

（3）向学生成绩数据库中插入如下一行数据：（**答案位置：光盘\TM\sl\16\11**）

```
14758  冯响 男 1990-4-1 计算机应用技术 50 NULL
"<联系方式 xmlns="http://mr.com"  姓名="冯响">
<email>FX@ins.net</email>
<电话>139××××××××</电话>
<地址 国家="中国">
      <邮政编码>453000</邮政编码>
      <省或直辖市>河南省</省或直辖市>
      <市或县>新乡</市或县>
      <详细地址>XX 路 X 号</详细地址>
</地址>
</联系方式>"
```

第17章

XQuery 技术的使用

（视频讲解：49 分钟）

DB2 V9 版本中有一个新特性，就是增加了 XML 类型数据存储，操作这种数据最理想的方式是通过 XQuery 来进行，XQuery 是 XML 数据检索的标准，可以在 W3C 网站上查看其规范，本章将详细讲解 XQuery 技术的使用。

通过阅读本章，您可以：

- 了解 XQuery 技术的基础知识
- 掌握如何使用 XQuery 技术查询 XML 数据
- 掌握如何使用 XQuery 技术更新 XML 数据

17.1 XQuery 基础

XQuery 是万维网联盟（W3C）设计的一种函数编程语言，用于满足查询和修改 XML 数据的特定需求。XQuery 用于遍历 XML 文档树并提取 XML 片段，XQuery 还包括一些表达式，这些表达式用于创建、操纵和遍历 XML 项序列及构建新的 XML 数据。

IBM 已扩展了 DB2 的所有主要应用程序编程接口（API），用来支持将 XQuery 作为 SQL 那样的一种语言。其中包括 CLI/ODBC、嵌入式 SQL、JDBC 和.NET，因此 DB2 命令行处理器和命令编辑器也支持 XQuery。

17.1.1 XQuery 查询的组成部分

在 XQuery 中，表达式是查询的主要构建块，此表达式可由使用运算符或关键字组合而成的多个 XQuery 表达式构成，表达式可以进行嵌套，用来组成查询的主体。在查询主体前面还可以具有序言，序言包含一系列用于定义查询处理环境的声明。

XQuery 查询结构图如图 17.1 所示。

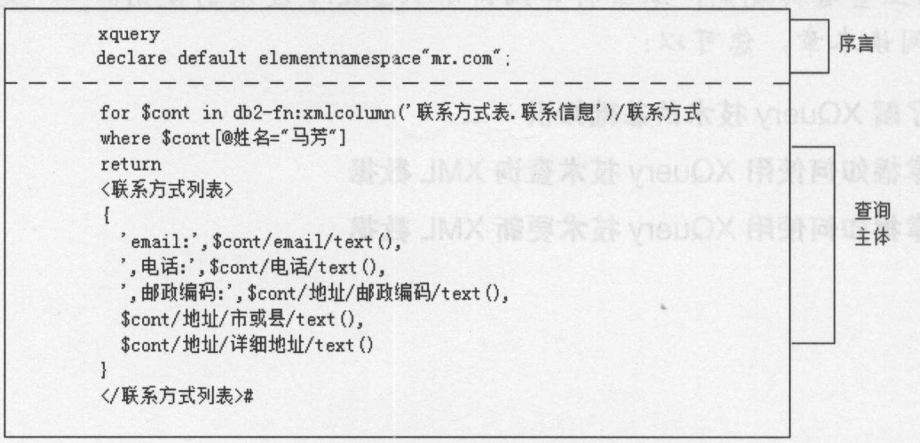


图 17.1 典型的 XQuery 查询的结构图

在此示例中，序言包含默认名称空间声明，它指定要用于无前缀元素名称和类型名的名称空间 URI：“http://mr.com”。

查询主体是一个用于构造：“联系方式表”元素的表达式。“联系方式表”元素的内容是姓名为“马芳”的学生的各种联系方式的集合。

执行该 XQuery 查询命令会返回：

```
<联系方式表 xmlns="http://mr.com"
email: MF@106.com,电话:176××××××××,邮政编码:120011,地址:广东省广州市 48 号>
</联系方式表>
```

17.1.2　基本规定

1．区分大小写

XQuery 是一种区分大小写的语言。

XQuery 中的关键字使用小写字符，并且不是保留关键字。XQuery 表达式中的名称可以与语言关键字相同。

2．注释

序言或查询主体中允许使用注释。注释不会影响查询处理。

注释由通过符号“(:”“:)”定界的字符串组成。

例如 XQuery 中的注释示例：

```
(: Build Your Dream :)
```

17.1.3　XML 名称空间和 QName

XML 名称空间可避免命名冲突。XML 名称空间是名称空间 URI 标识的名称的集合。名称空间提供了一种方法来限定用于 XQuery 中的元素、属性、数据类型和函数的名称。使用名称空间前缀限定的名称为限定名称（QName）。

1．限定名称（QName）

QName 由可选名称空间前缀和局部名组成。名称空间前缀与局部名用冒号隔开。如果名称空间前缀存在，那么会绑定至 URI（通用资源标识），并且会提供 URI 的简单形式。例如“xmlns:电话”是一个 QName，其中“xmlns”是绑定到 URI：“http://mr.com”的前缀。

如果两个 QName 具有相同的名称空间 URI 和局部名，那么表示这两个 QName 相等。这意味着即使两个 QName 的前缀不同，只要它们的前缀绑定至同一名称空间 URI，那么这两个 QName 也是匹配（相等）的。

QName（限定名称）遵循“W3C Recommendation Namespaces in XML”中定义的语法。

2．静态已知名称空间

名称空间声明会将名称空间前缀绑定至 URI，用于控制 QName 在查询表达式中解释的名称空间绑定集合称为静态已知名称空间。静态已知名称空间是查询表达式的属性，与表达式处理的数据无关。

某些名称空间前缀是预先声明的，其他名称空间则可通过查询序言或元素构造函数中的声明进行添加。

（1）预先声明名称空间。

表 17.1 描述了 DB2 XQuery 中的预先声明名称空间前缀。

表 17.1　DB2 XQuery 中的预先声明名称空间

前　缀	URI	描　述
xml	http://www.w3.org/XML/1998/namespace	XML 保留名称空间
xs	http://www.w3.org/2001/XMLSchema	XML 模式名称空间
xsi	http://www.w3.org/2001/XMLSchema-instance	XML 模式实例名称空间
fn	http://www.w3.org/2005/xpath-functions	默认函数名称空间
xdt	http://www.w3.org/2005/xpath-datatypes	XQuery 类型名称空间
db2-fn	http://www.ibm.com/xmlns/prod/db2/functions	DB2 函数名称空间

（2）在查询序言中声明，通过使用名称空间声明或默认名称空间声明。

（3）由元素构造函数中的名称空间声明属性声明。

（4）由 SQL/XML 提供。

SQL/XML 可提供下列一组名称空间：

☑ SQL/XML 预先声明名称空间。

☑ 在 SQL/XML 构造函数和其他 SQL/XML 表达式中声明的名称空间。

注意

（1）元素构造函数中的名称空间声明属性可能覆盖序言中声明的名称空间。

（2）可通过在查询序言中指定名称空间声明来覆盖预先声明的名称空间前缀。

（3）元素构造函数中的后续名称空间声明或序言中的名称空间声明属性可能覆盖 SQL/XML 提供的名称空间。

17.1.4　序言

序言是一系列声明，用于定义查询的处理环境。序言中的每个声明后面使用分号（;）分隔。序言是查询的可选部分，有效查询可由不带序言的查询主体组成。

XQuery 序言的语法为：

```
[Version declaration]
[Namespace declaration]
[Boundary-space declaration]
[Construction declaration]
[Copy-namespaces declaration]
[Empty order declaration]
[Default element/type namespace declaration]
[Default function namespace declaration]
[Ordering mode declaration]
```

其中：

```
Version declaration :: = xquery version "1.0" [encoding StringLiteral];
Namespace declaration :: = declare namespace prefix = URILiteral;
```

```
Boundary-space declaration :: = declare boundary-space strip | declare boundary-space preserve;
Construction declaration :: = declare construction preserve;
Copy-namespaces declaration :: = declare copy-namespaces preserve,inherit;
Default element/type namespace declaration ::= declare default element namespace URILiteral;
Default function namespace declaration ::= declare default function namespace URILiteral;
Empty order declaration ::= declare default order empty greatest;
Ordering mode declaration ::= declare ordering unordered;
```

注意

（1）每个声明只能指定一次，但名称空间声明除外。

（2）斜体字表示需要用某些字符串代替。

参数说明如下。

- ☑ 序言中包括可选版本声明、名称空间声明和 Setter，Setter 是设置影响查询处理的属性值的可选声明。但是，因为 DB2 XQuery 中的 Setter 在每种情况下只支持一个选项，所以它们不会更改处理环境。如果版本声明存在，那么它必须出现在序言开头。Setter 及其他声明可按任意顺序出现在序言版本声明之后的位置。
- ☑ 声明版本：DB2 XQuery 唯一支持的版本为“1.0”。StringLiteral：指定表示编码名称的字符串文字，DB2 XQuery 总是假定编码为 UTF-8。
- ☑ 名称空间声明：查询序言中的名称空间声明会声明名称空间前缀并使该前缀与名称空间 URI 相关联。前缀与名称空间 URI 之间的关联称为“名称空间绑定”。在名称空间声明中绑定的名称空间将添加至“静态已知名称空间”。“静态已知名称空间”包含处理查询期间可用于解析名称空间前缀的所有名称空间绑定。
- ☑ 除非名称空间声明被直接元素构造函数中的名称空间声明属性覆盖，否则此声明在声明查询的作用域内。如果查询序言中存在同一名称空间前缀的多个声明，那么会产生错误。
  - ➢ prefix：指定绑定值 URILiteral 指定的 URI 的名称空间前缀。该名称空间前缀在限定名称（QName）中用于标识元素、属性、数据类型或函数的名称空间。
  - ➢ URILiteral：指定前缀绑定至的 URI。URILiteral 必须为包含有效 URI 的非零长度文字串。
- ☑ 边界空格声明：边界空格策略会控制元素构造函数对空格的处理方式。边界空格包括元素构造函数中的标记或带括号表达式之间的边界中自己出现的所有空格字符。
  - ➢ Strip：指定构造元素时除去边界空格。
  - ➢ Preserve：指定构造元素时保留边界空格。
- ☑ 构造声明：构造方式控制如何对复制的元素和属性节点指定类型注释，以形成新的构造节点。
  - ➢ Preserve：指定构造元素的被复制属性和后代将保留其原始类型。
- ☑ 复制名称空间声明：复制名称空间方式控制元素构造函数复制现有元素节点时指定的名称空间绑定。
  - ➢ Preserve：指定原始元素的名称空间作用域将保留在新副本中。
  - ➢ Inherit：指定复制的节点将继承构造节点的名称空间作用域。
- ☑ 默认元素/类型名称空间声明：指定名称空间，该名称空间用于元素和类型名称的无前缀的限

定名称。

- Element：指定声明为默认元素/类型名称空间声明。
- URILiteral：指定表示名称空间的 URI 的字符串文字。字符串文字必须是有效 URI 或零长度字符串。如果默认元素/类型名称空间声明中的字符串文字是零长度字符串，那么无前缀元素和类型名称不在任何名称空间内。

☑ 默认函数名称空间声明：指定用于函数调用中无前缀函数名的名称空间 URI。

- Function：指定该声明是默认函数名称空间声明。
- URILiteral：指定表示名称空间的 URI 的字符串文字。字符串文字必须是有效 URI 或零长度字符串。如果默认函数名称空间声明中的字符串文字为零长度字符串，那么所有函数调用必须使用加前缀函数名，其原因是每个函数都在同一个名称空间中。

☑ 空排序声明：在处理 FLWOR 表达式的 order by 子句时，查询序言中的空排序声明用于控制空序列或 NaN 值是解释为最高值还是最低值。

- Greatest：处理 FLWOR 表达式中的 order by 子句时空序列总是解释为最高值。NaN 值解释为大于空序列以外的所有的其他值。

☑ 排序方式声明：排序方式会定义查询结果中的节点排序。

- Unordered：指定“XQuery1.0：An XML Query Language”中的排序方式规则不起作用。

17.1.5 表达式

表达式是查询的基本构建块。表达式可以单独使用，也可和其他表达式组合使用。DB2 XQuery 支持使用多种表达式来处理 XML 数据。如主表达式、路径表达式、谓词、序列表达式、算术表达式、比较表达式、逻辑表达式、构造函数、FLWOR 表达式、条件表达式、定量表达式、数据类型转换表达式、变换表达式和更新表达式等。

1. 主表达式

主表达式是语言的基本主体。主表达式包括文字、变量引用、带括号表达式、上下文项表达式、构造函数和函数调用。

文字是原子值的直接语法表示。DB2 XQuery 支持两种类型的文字：数字文字和字符串文字。数字文字是类型为 xs:integer、xs:decimal 或 xs:double 的原子值。字符串文字是类型为 xs:string 的原子值，括在定界单引号（‘’）或双引号（“”）中。字符串文字可包括预定义实体引用和字符引用。

【例 17.1】 下列字符串文字都是有效字符串文字。

```
"13.8"
"What should I do?"
"Ben &amp;amp; Jim&amp;apos;s"
```

“预定义实体引用”是一个简短字符序列，表示在 DB2 XQuery 中具有一定语法意义的字符。预定义实体引用以和号（&）开头，以分号（;）结尾。预定义实体引用所表示的字符如表 17.2 所示。处理字符串文字时，每个预定义实体引用将替换为它所表示的字符。

表 17.2　DB2 XQuery 中的预定义实体引用

实 体 引 用	表示的字符
<	<
>	>
&	&
"	“
'	‘

“字符引用”是对 Unicode 字符的 XML 样式引用，Unicode 字符由其十进制或十六进制代码点标识。字符引用以“&#x”或“&#”开头，以分号（;）结尾。如果字符引用以“&#x”开头，那么终止分号（;）之前的数字和字母将根据 ISO/IEC 10646 标准表示为字符代码点的十六进制形式。如果字符引用以“&#”开头，那么终止分号（;）之前的数字将标识为字符代码点的十进制形式。

例如：字符引用“□”或“€”表示欧元符号（€）。

“变量引用”是将美元符号（$）用作前缀的 NCName（数据类型 xs:NCName 表示不使用冒号的 XML 名称），对查询求值时，每个变量引用会解析为绑定至该变量的值。每个变量引用必须与变量作用域中引用位置处的名称相匹配。

变量名称只能在 FLWOR 表达式中声明一次，后面将讲到 FLWOR 表达式。

【例 17.2】　下面语句的返回结果是什么？

```
xquery for $i in (1,2)
for $i in ("a","b")
return $i
```

上面的语句是错误的，DB2 XQuery 不支持以上表达式。如果变量引用与作用域中的两个或多个变量绑定相匹配，那么该引用会引用内部绑定，即作用域更小的绑定。

【例 17.3】　下面语句的返回结果是什么？

```
xquery let $seq :=(10,20,30)
return $seq[2]
```

用 FLWOR 表达式将变量$seq 绑定至序列（10,20,30），返回的值为 20。

上下文项表达式由单个句点字符（.）组成。上下文项表达式会求值为当前正在处理的项，又称为上下文项。上下文项可以是节点或原子值，只能在路径表达式和谓词表达式中定义上下文项。

【例 17.4】　下面语句的返回结果是什么？

```
xquery (1 to 100)[. Mod 5 eq 0]
```

将生成 1~100 之间可被 5 整除的整数序列，结果集是 20 个整数组成的序列。

函数调用由 QName 组成，后跟用圆括号括起来的零个或多个表达式（称为自变量）。

DB2 XQuery 支持调用内置 DB2 XQuery 函数和特定于 DB2 的函数。内置 XQuery 函数在绑定至前缀 fn 的名称空间 http://www.w3.org/2005/xpath-functions 中。特定于 DB2 的函数在绑定至前缀 db2-fn 的名称空间 http://www.ibm.com/xmlns/prod/db2/functions 中。除非名称空间被查询序言中的默认函数声明覆盖，否则默认函数名称空间是内置 XQuery 函数的名称空间（绑定至前缀 fn）。

下面的步骤说明 DB2 XQuery 对函数求值时所用的过程。

实现步骤如下：

（1）DB2 XQuery 对函数调用中作为自变量传递的每个表达式求值，并返回每个表达式的值。

（2）对每个自变量返回的值将转换为该自变量应使用的数据类型。如果自变量应该为原子值或原子值序列，那么 DB2 XQuery 会使用下列规则将自变量的值转换为预期类型。

- ☑ 将对给定的值进行原子化，这会产生原子值序列。
- ☑ 原子序列中类型为 xdt:untypedAtomic 的数据类型转换为预期原子类型。对于应该为原子自变量的内置函数，类型为 xdt:untypedAtomic 的自变量的数据类型将转换为 xsdouble。
- ☑ 将对原子序列中任何可提升至预期原子类型的数字项应用数字类型提升。数字项包括类型为 xs:integer（或派生自 xs:integer）、xs:decimal、xs:float 或 xs:double 的项。
- ☑ 如果预期类型为 xs:string，那么原子序列中类型为 xs:anyURI（或派生自 xs:anyURI）的每项将提升至 xs:string。

（3）将使用函数自变量的转换值对函数求值。函数调用将生成函数声明的返回类型的实例，或者产生错误。

使用字符串自变量的函数调用类型为 xs:string 的自变量，并返回类型为 xs:string 的值。

例如，函数 fn:upper-case（xs:string_expression）将类型为 xs:string 的自变量 xs:string_expression 转换为大写形式的类型为 xs:string 的值并返回。函数 db2-fn:xmlcolumn()中，参数是模式中的列名。

例如，db2-fn:xmlcolumn('MR.联系方式表.联系信息')。

注意

（1）db2-fn:xmlcolumn('MR.联系方式表.联系信息')中的表模式名“MR”必须大写，因为 db2-fn:xmlcolumn 函数对大小写敏感。

（2）由于 DB2 的语句可能比较复杂，语句中又包含子语句，而子语句一般用“;”结尾，所以整个语句的终止字符不能再采用“;”，本书采用“#”。如果语句均为简单语句则仍然可以用“;”结尾。

这并不是说只能以“#”作为复杂语句终止字符，事实上，“#”、“@”、“!”、“$”、“%”、“*”、“+”、“\”、“&”、“?”和空格等都可以作为复杂语句终止字符（但其中某些语句终止字符容易与 DB2 语句中的某些符号混淆），最常用的语句终止字符是“;”、“#”、“@”和“!”。虽然 DB2 帮助文档中经常以“@”作为语句终止字符，但考虑到 DB2 V9 以上版本支持 XML 数据，且引用 XML 文档中的属性的格式为“@[属性名]”，所以为避免混淆，本书都使用“#”作为语句终止字符。

语句终止字符在命令编辑器中有两种方式可以进行设置。

第一种：将命令编辑器左下角的“语句终止符号”后面的文本框中的“;”改成指定的语句终止字符。这种方法只对当前命令编辑器中的语句有效。

第二种：在控制中心或命令编辑器中，选择“工具”主菜单，然后选择“工具设置”子菜单。在弹出的“工具设置”对话框中打开“常规”选项卡，在“常规”选项卡中选中“使用语句终止字符”复选框，并在该行的文本框中输入指定的语句终止字符，最后单击“工具设置”对话框右上角的“×”按钮关闭该对话框。这种方法指定的语句终止字符将成为命令编辑器的默认设置。

使用序列自变量的函数调用实例如例 17.5 所示。

【例 17.5】　以下语句运行的返回值是什么？

```
xquery fn:max((1,2,3))#
```

函数 fn:max((1,2,3))采用序列(1,2,3)作为单个自变量。因为函数 fn:max 需要原子值序列形式的单个自变量，所有需要嵌套圆括号，返回的值为 3。

2．谓词

谓词会通过保留限定项来过滤序列。谓词由表达式组成，该表达式会用方括号（[ ]）括起来，称为“谓词表达式”。谓词表达式会将所有选项作为上下文项对序列中的每一项进行一次求值。谓词表达式的每次求值将返回 xs:boolean 值，称为“谓词布尔值”。谓词布尔值为 true 的项会“保留”下来，而谓词布尔值为 false 的项会被“废弃”。

下面的规则用于确定谓词布尔值：

☑ 如果谓词表达式返回非数字值，那么谓词布尔值为谓词表达式的有效布尔值。

☑ 如果谓词表达式返回数字值，那么只有在序列中所处位置等于该数值的项的谓词布尔值才为 true，对于其他项，谓词布尔值为 false，此类谓词称为“数字谓词”或“位置谓词”。

注意

（1）仅当序列具有确定顺序时，使用数字谓词从序列中选择的项才是确定的。

（2）谓词的行为取决于谓词表达式是否返回数字值，这可能无法通过观察谓词表达式进行判断。可通过使用 fn:boolean 函数强制谓词使用有效布尔值，格式为：“[fn:Boolean(PredicateExpression)]”。此外，还可通过使用 fn:position 函数强制谓词按位置谓词处理，格式为“[fn:position() eq PredicateExpression]”。

【例 17.6】　解释下列表达式的含义。

```
$products[5]
descendant::toy[@color="Green"]
employee[secretary][assistant]
(<cat />,<fish />,58,<zebra />)[2]
```

参数说明如下。

☑ $products[5]中，数字谓词[5]表示只保留绑定至变量$products 的序列中的第 5 项。

☑ descendant::toy[@color="Green"]会选择充当元素 toy 并且颜色属性值为“Green”的所有上下文节点后代。

☑ employee[secretary][assistant]会选择同时具有 secretary 子代元素和 assistant 子代元素的所有上下文节点职员子代。

☑ （<cat />,<fish />,47,<zebra />）[2]会返回元素<fish />。

3. FLWOR 表达式的语法

FLWOR 表达式的语法格式为：

```
[for_clause | <let_clause>] [where Expression] [order_by_clause] <return Expression>
```

其中：

```
for_clause::= for $VariableName [at $PositionalVariableName] in Expression
let_clause::=let $VariableName :=Expression
order_by_clause ::=order by Expression [ascending | descending]
```

参数说明如下。

- ☑ 在 XQuery 语句中，FLWOR 表达式由 for、let、where、order by 和 return 子句组成，所有这些子句都是可选的。
- ☑ Expression：XQuery 表达式。
- ☑ Return：如果 return 子句包含非更新表达式，那么 FLWOR 表达式为非更新表达式，对 return 子句求值的所有结果将并置到单个序列中，此序列是 FLWOR 表达式生成的。如果 return 子句包含更新表达式，那么 FLWOR 表达式是更新表达式，必须在变换表达式的 modify 子句中指定 FLWOR 更新表达式，FLWOR 更新表达式会生成更新列表，在将变换表达式的 modify 子句中的其他更新表达式返回的其他更新合并后，包含更换的表达式会执行更新。

4. 路径表达式

路径表达式表示 XML 树中的节点。DB2 XQuery 中的路径表达式以 XPath20 的语法为基础。表达式可用轴步骤开头，也可用斜杠或双斜杠字符开头。最终步骤之前的每个步骤会生成节点序列，这些节点将用作后续步骤的上下文节点。

路径表达式的语法结构为：

```
{[/ | //]<轴步骤 | 过滤表达式>} [,...n]
```

参数说明如下。

- ☑ 路径表达式包含用斜杠（/）或双斜杠（//）字符隔开的一个或多个步骤。
- ☑ 路径表达式的每个步骤是轴步骤或过滤表达式。
- ☑ 初始斜杠（/）指示路径起始于包含上下文节点的树的根节点，该节点必须是文档节点。路径表达式中的斜杠字符用于隔开步骤。由多个斜杠和元素（或属性）组成的路径表达式表示每个约束或属性在 XML 文档中从根节点开始的绝对路径。
- ☑ 初始双斜杠字符（//）指示路径起始于初始节点序列，该序列由包含上下文节点的树的根节点及其所有后代组成，根节点必须是文档节点。路径表达式中的双斜杠字符也用于隔开步骤。路径表达式中出现双斜杠表示双斜杠后面的所有节点或属性都包含在双斜杠前面的节点的所有后代组成的子树中，也就是说，双斜杠（//）表示元素在 XML 文档中的相对路径。

路径表达式中“过滤表达式”的语法如下：

```
过滤表达式::=PrimaryExpression[PredicateExpression]
```

参数说明如下。

☑ 过滤表达式由后跟零个或多个谓词的主表达式组成。

☑ PredicateExpression：谓词，用于确定是保留还是废弃序列项的表达式。

☑ PrimaryExpression：主表达式。

路径表达式中的“轴步骤”的语法如下：

```
轴步骤::= [axis::] node_test[PredicateExpression]
```

参数说明如下。

☑ 轴步骤返回可通过指定轴从上下文节点到达的节点的序列。轴步骤的结果总是零个或多个节点的序列。

☑ 轴步骤由 3 部分组成：轴（axis）、节点测试（node_test）及零个或多个谓词（Predicate Expression）。轴用来指定移动方向，节点测试用于选择节点的条件，而谓词用来过滤步骤返回的序列。

接下来解释一下轴和节点测试。

☑ 轴（axis）

- 轴表示在 XML 文档或片段中移动的方向。轴步骤可以是正向步骤或逆向步骤。正向步骤从上下文节点开始并且按文档顺序在 XML 树中移动，而逆向步骤从上下文节点开始并且按逆向文档顺序在 XML 树中移动。如果上下文项不是节点，那么表达式会产生错误。
- 受支持的轴包括：子代、后代、属性、自身、后代或自身（以上 5 项为正向）和父代（逆向）。轴步骤的非缩写语法包括用两个冒号隔开的轴名称和节点测试，并后跟零个或多个谓词。可通过省略轴并使用速记表示法来缩写轴表达式的语法。

☑ 节点测试（node-test）

- 节点测试是一个条件，对于轴步骤选择的每个节点，此条件必须为 true。此测试可以是根据节点名称选择节点的名称测试，或者是根据节点种类选择节点的种类测试。
- 名称测试由 QName 或通配符组成。

表 17.3 中描述的是在 DB2 XQuery 中支持的名称测试。

表 17.3　DB2 XQuery 中支持的名称测试

测　试	描　述
QName	匹配 QName 等于指定 QName 的任何节点（在指定轴上）。如果该轴为属性轴，那么此测试与属性节点相匹配。在所有的其他轴上，此测试与元素节点相匹配
*	匹配指定轴上的所有节点。如果该轴为属性轴，那么此测试与所有属性节点相匹配。在所有其他轴上，此测试与所有元素节点相匹配
NCName:*	指定表示 QName 的前缀部分的 NCName。此名称测试与指定轴上的某些节点相匹配，这些节点的名称空间 URI 与前缀绑定至的名称空间 URI 相匹配。如果该轴为属性轴，那么此测试与属性节点相匹配。在所有其他轴上，此测试与元素节点相匹配
*:NCName	指定表示 QName 的局部的 NCName。此名称测试与指定轴上的某些节点相匹配，这些节点的局部名等于 NCName。如果该轴为属性轴，那么此测试与属性节点相匹配。在所有其他轴上，此测试与元素节点相匹配

表 17.4 中描述的是在 DB2 XQuery 中支持的种类测试。

表 17.4　DB2 XQuery 中支持的种类测试

测　试	描　述
Node()	匹配指定轴上的任何节点
Text()	匹配指定轴上的任何文本节点
Comment()	匹配指定轴上的任何注释节点
Processing-instruction()	匹配指定轴上的任何处理指定节点
Element() 或 Element(*)	匹配指定轴上的任何元素节点
Attribute() or Attribute(*)	匹配指定轴上的任何属性节点
Document-node()	匹配指定轴上的任何文档节点

路径表达式中的轴的缩写语法：XQuery 提供了缩写语法，以表达路径表达式中的轴。表 17.5 描述在 DB2 XQuery 中轴的缩写语法和示例。

表 17.5　路径表达式的缩写语法

缩写语法	描　述	示　例
未指定轴	child::的速记缩写，但轴步骤指定节点测试的 attribute()除外。轴步骤指定属性测试时，省略轴是 attribute::的速记形式	路径表达式 section/para 是 child::section/child::para 的缩写；路径表达式 section/attribute()是 child::section/attribute::attribute()的缩写
@	attribute::的速记缩写	路径表达式 section/@id 是 child::section/attribute::id 的缩写
//	/descendant-or-self::node()的速记缩写，但此缩写出现在路径表达式开头时除外。此缩写出现在路径表	路径表达式 div1//para 是 child::div1/descendant-or-self::node()/
../	parent::node()的速记缩写	路径表达式../title 是 parent::node()/child::title 的缩写

表 17.6 描述在 DB2 XQuery 中轴的非缩写语法和缩写语法的对比示例。

表 17.6　DB2 XQuery 中轴的非缩写语法和缩写语法的对比示例

非缩写语法	缩写语法	含　义
Child::para	Para	选择充当上下文节点子代的 para 元素
Child::*	*	选择充当上下文节点子代的所有元素
Child::text()	text()	选择充当上下文节点子代的所有文本节点
Child::node()	Node()	选择上下文节点的所有子代。因为属性未被视作节点的子代，所以此表达式不返回任何属性节点
Attribute::name	@name	选择上下文节点的 name 属性
Attribute::*	@*	选择上下文节点的所有属性
Child::para[fn:position()=1]	Para[1]	选择充当上下文节点子代的第一个 para 元素
Child::para[fn:position()=fn:last()]	Para[fn:last()]	选择充当上下文节点子代的最后一个 para 元素
/child::book/child::chapter[fn:position()=5]/child::section[fn:position()=2]	/book/chapter[5]/section[2]	选择该书第 5.2 节，其父代包含上下文节点的文档节点

续表

非缩写语法	缩 写 语 法	含　义
Child::para[attribute::type="warning"]	Para[@type="warning"]	选择上下文节点的类型属性值为 warning 的所有 para 子代
Child::para[attribute::type='warning'][fn:position()=5]	Para[@type="warning"][5]	选择上下文节点的类型属性值为 warning 的第 5 个 para 子代
Child::para[fn:position()=5][attribute::type="warning"]	Para[5][@type="warning"]	选择上下文节点的类型属性值为 warning 的第 5 个 para 子代
Child::chapter[child::title='Introduction']	Chapter[title="Introduction"]	选择上下文节点的 chapter 子代，该子代的一个或多个 title 子代的类型值等于字符串 Introduction
Chilid::chapter[child::title]	Chapter[title]	选择上下文节点的 chapter 子代，该子代具有一个或多个 title 子代

【例 17.7】“$cont/地址/邮政编码/text()”、“$cont//邮政编码/text()”、“$cont/地址//邮政编码/text()”、“$cont//地址/邮政编码/text()”和“$cont//地址//邮政编码/text()”都可以查找到“联系方式表”的“联系信息”列中的某个 XML 文档中“邮政编码”节点下的文本。其中，$count 指代“联系方式表”的“联系信息”列的“联系方式”节点。它们有什么区别？

（1）“$cont/地址/邮政编码/text()”表示从 XML 文档中“$cont”根节点开始一级一级查找“邮政编码”节点，“地址”节点是“$cont”节点的子代，“邮政编码”节点是“地址”节点的子代。

（2）“$cont//邮政编码”表示“邮政编码”节点在根节点“$cont”的所有后代形成的树结构中。

（3）“$cont/地址//邮政编码”表示“地址”节点是根节点“$cont”的子代，“邮政编码”节点在“地址”节点的所有后代形成的树结构中。

（4）“$cont//地址/邮政编码”表示“地址”节点在根节点“$cont”的所有后代形成的树结构中，“邮政编码”节点是“地址”节点的子代。

（5）“$cont//地址//邮政编码”表示“地址”节点在根节点“$cont”的所有后代形成的树结构中，“邮政编码”节点在“地址”节点的所有后代形成的树结构中。

5. 数据类型转换表达式

数据类型转换表达式会根据现有值创建特定类型的新值。

（1）使用 CAST 转换成数据类型。

使用 CAST 转换数据类型的转换表达式有两个操作数：输入表达式和目标类型。对数据类型转换表达式求值时，原子化用于将输入表达式的结果转换为原子值或空序列。如果原子化会生成包含多个原子值的序列，那么会返回错误。如果未返回错误，那么数据类型转换表达式会尝试根据输入值创建新的目标类型值。数据类型转换不支持输入和目标类型的某些组合。只有当目标类型后跟问号（?）时，空序列才是有效的输入值。

CAST 语句语法为：

```
Expression cast as TargetType[?]
```

参数说明如下。

☑ Expression：返回单个原子值或空序列的 XQuery 表达式。仅当 TargetType 后跟问号（?）时，

才允许使用空序列。

☑ TargetType：Expression 的值将转换至的类型。TargetType 必须是属于预定义原子 XML 模式类型的原子类型。数据类型 xs:NOTATION、xdt:anyAtomicType 和 xs:anySimpleType 对于 TargetType 是无效类型。

☑ ?：指示 Expression 的结果可能为空序列。

（2）使用 XMLCAST 函数将 XMLQUERY 子句的结果转换为非 XML 类型。

XMLCAST 函数语法为：

```
XMLCAST（<XMLQUERY_expression | NULL |parameter-marker>AS data-type）
```

说明

（1）XMLCAST 函数将 XML 值转换为非 XML 数据类型，或者将非 XML 值转换为 XML 值。例如，将 SQL/XML 值转换为 SQL 数据类型。假设转换的源来自 XML 模式原子类型或无类型文本，或者是值为原子类型或无类型文本的节点。如果想要将 XMLQUERY 函数的结果返回至 SQL 上下文以进行进一步处理（如进行比较或排序操作），那么需要将返回的 XML 值增强类型转换为兼容的 SQL 类型。

（2）指定 XMLCAST 能够在 XML 值和非 XML 值之间进行强制类型转换。

注意

只有当在 XMLQUERY 中指定的 XQuery 表达式返回的序列包含一个已打散的项（能够转换为 SQL 原子类型的项）时，才能将 XMLQUERY 的结果强制类型转换为 SQL 数据类型。

17.2 查询 XML 数据

有两种方式可查询 XML 数据，一种是使用界面查询表中的 XML 数据，另一种是使用普通 SQL 查询 XML 数据，下面分别进行介绍。

17.2.1 界面查询表中 XML 数据

【例 17.8】 在界面中查询“联系方式表”的“联系信息”列的 XML 数据。

实现步骤如下：

（1）在“控制中心”打开“联系方式表”。单击“联系信息”中 XML 后面的“...”按钮，如图 17.2 所示。

（2）弹出如图 17.3 所示的对话框，在该对话框中即可显示相应的 XML 文档信息。

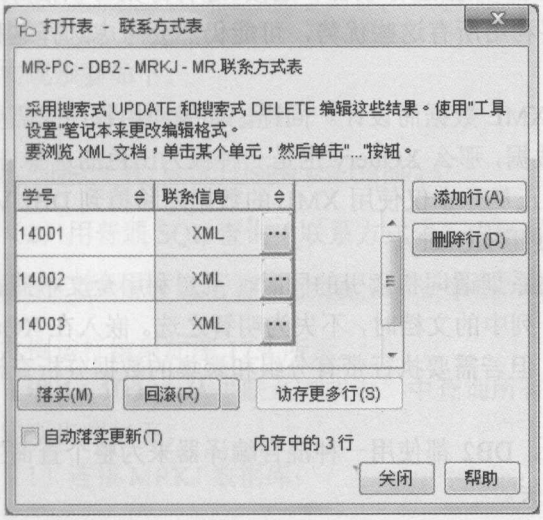


图 17.2　“打开表-联系方式表”对话框

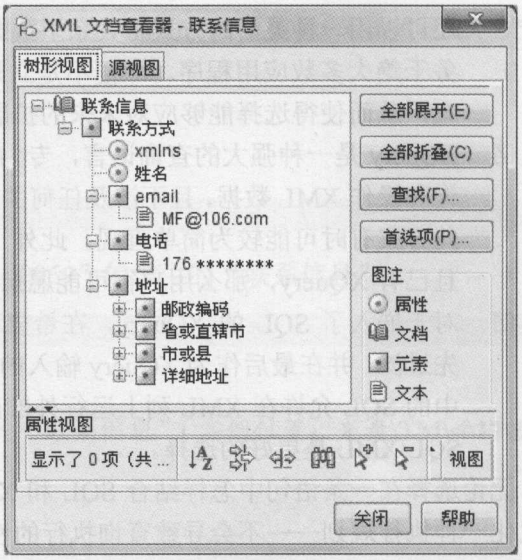


图 17.3　查看信息

17.2.2　如何选择查询 XML 数据的语言

DB2 中的 PureXML 支持为管理 XML 数据提供了高效且通用的方法。DB2 以 XML 数据自身固有的分层格式存储和处理这些数据，避免因为将 XML 存储为 CLOB 中的文本或将它映射为关系表而导致的性能和灵活性限制。与只使用 XML 的数据库不同，DB2 V9 还提供了关系型数据与 XML 数据在数据库中的无缝集成——甚至是表的某一行中的集成。这样的灵活性表现在语言支持中，实现了可访问关系型数据、XML 数据，或者同时访问这两种数据。

DB2 允许通过以下 5 种可选方法中的任一种查询 XML：

- ☑ 普通 SQL（不含 XQuery）。
- ☑ 全文本搜索（需要在 DB2 中安装 NET SEARCH EXTENDER 插件，且全文本搜索属于 SQL 但不属于 PureXML）。
- ☑ SQL/XML，即嵌入了 XQuery 的 SQL。
- ☑ XQuery 作为独立语言（不含普通 SQL 和全文搜索）。
- ☑ 嵌入了 SQL 的 XQuery。

可以在普通 XQuery、SQL/XML 或嵌入了 SQL 的 XQuery 中表达许多对 XML 数据的查询。在特定情况下，可能会发现其中之一能够比其他方案更为直观地表达查询逻辑。一般而言，查询 XML 的“正确”途径需要在“逐个处理”的基础上加以选择，需要考虑应用程序的具体需求和特征。

指导原则总结如下：

- ☑ 不带任何 XQuery 的普通 SQL 仅对 XML 全文档检索以及整个文档的插入、删除、更新操作有用。文档的选择必须基于同一表中的非 XML 列。
- ☑ 带有嵌入在 SQL 内的 XQuery 语句的 SQL/XML 功能性强且局限性少。可在 XML 列上表示谓词、提取文档片段、向 XQuery 表达式传递参数标记、使用全文本搜索及 SQL 级聚集与分组，

还可以用一种灵活的方式将关系型数据与 XML 数据进行联合和连接。这种方案可以很好地服务于绝大多数应用程序。即便不是立即需要利用所有这些优势，可能仍然会考虑选择这种方案，从而使得选择能够应对未来的扩展。

- ☑ XQuery 是一种强大的查询语言，专为查询 XML 数据而设计。同样，如果应用程序只需要查询和操作 XML 数据，且不涉及任何关系型数据，那么 XQuery 也是一种极为出色的选择方案。此方案有时可能较为简单直观。此外，如果正从一个仅使用 XML 的数据库移植到 DB2 V9，且已有 XQuery，那么用户很可能愿意继续使用 XQuery。
- ☑ 对于嵌入了 SQL 的 XQuery，在希望利用关系型谓词和索引的同时，又想利用全文本搜索预先过滤，并在最后作为 XQuery 输入的 XML 列中的文档时，不失为明智之选。嵌入在 XQuery 中的 SQL 允许在 XML 列上运行外部函数，但若需要执行带有分组和聚集的数据分析查询，SQL/XML 是更好的选择。

无论选择在一条语句中怎样结合 SQL 和 XQuery，DB2 都使用一种混合编译器来为整个查询生成及优化一种执行规划——不会导致查询执行的性能损失。

表 17.7 中总结了查询 XML 数据的 4 种不同可选方案各自的性能。

表 17.7　查询 XML 数据的 4 种不同可选方案各自的性能

XML 数据	普通 SQL	SQL/XML	普通 XQuery	嵌入了 SQL/XML 的 XQuery
XML 谓词	-	++	++	++
关系谓词	++	++	-	+
XML 及关系谓词	-	++	-	++
将 XML 与关系型相连接	-	++	-	++
将 XML 与 XML 相连接	-	+	++	++
转换 XML 数据	-	o	++	++
插入、更新和删除	++	++	-	-
参数标记	+	++	-	-
全文搜索	+	++	-	++
XML 聚集与分组	-	++	o	o
函数调用	++	++	-	++

表 17.7 中，“-”表示给定语言不支持某项特性；“+”表示支持此特性，但存在更有效或更便捷的方式；“++”表示给定语言极为适合表现该特性；“o”表示尽管可表现此特性，但从某种程度上说，效果非常糟糕或者效率很低。

17.2.3　用普通 SQL 查询 XML 数据

可使用不带任何 XQuery 的普通 SQL 在不使用 XML 谓词的情况下读取整个文档。只要应用程序能够识别出 XML 文档，从而基于同一表中的关系谓词进行全文档搜索，那么这种方法就是一种非常简单易行的方法。

【例 17.9】　从“联系方式表”中查询学号为“140969”的学生的“联系信息”（**实例位置：光盘\TM\sl\17\1**）。

实现步骤如下。

（1）连接 MRKJ 数据库：

```
CONNECT TO MRKJ USER MR USING 1;
```

（2）用普通 SQL 查询“联系方式表”中学号为“140969”的学生的“联系信息”：

```
select 联系信息 from 联系方式表 where 学号='140969';
```

本例运行结果如图 17.4 所示。

【例 17.10】　从“联系方式表”中查询所有学生的“联系信息”（**实例位置：光盘\TM\sl\17\2**）。

实现步骤如下。

（1）连接 MRKJ 数据库：

```
CONNECT TO MRKJ USER MR USING 1;
```

（2）用普通 SQL 查询所有学生的“联系信息”：

```
select 联系信息 from 联系方式表;
```

本例运行结果如图 17.5 所示，单击“...”按钮便可查看信息。

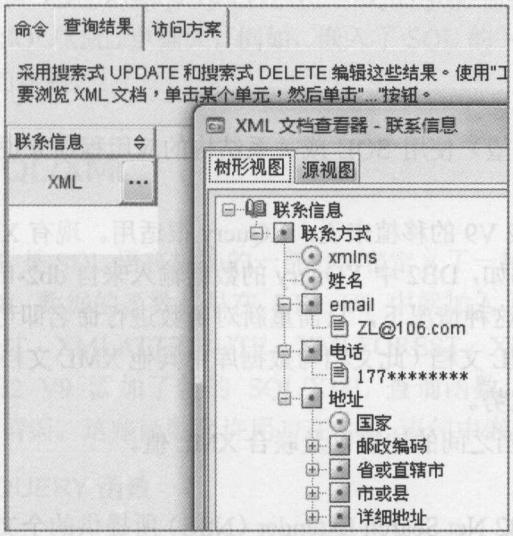


图 17.4　查询结果

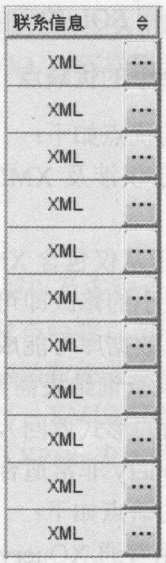


图 17.5　查询结果

该方法显而易见的缺陷就是无法表示关于 XML 数据本身的谓词，也无法只检索 XML 文档的片段。如果用户的查询不需要在 XML 上应用谓词，并且总是返回整个 XML 文档，那么普通 SQL 足以满足需要。在本例中，如果在 CHARACTER、VARCHAR 或 CLAB 列中存储 XML，对于全文档插入及检索操作而言，反而会带来一定的性能优势。

17.2.4 XQuery 和嵌入了 SQL 的 XQuery

在 XQuery 中，查询可以调用下列函数中的任意一种来获取 DB2 数据库中输入的 XML 数据：db2-fn:sqlquery 和 db2-fn:xmlcolumn。db2-fn:xmlcolumn 函数将检索整个 XML 列，而 db2-fn:sqlquery 将检索基于 SQL 全查询的 XML 值。使用 db2-fn:xmlcolumn 函数可以将 XQuery 作为独立语言查询 XML 数据，要使用嵌入了 SQL 的 XQuery，需要 db2-fn:sqlquery 函数。

1．db2-fn:xmlcolumn 函数和 db2-fn:sqlquery 函数

db2-fn:xmlcolumn 函数：db2-fn:xmlcolumn 函数采用一个字符串文字自变量，用于标识一个表或视图中的 XML 列，并返回该列中的 XML 值序列。此函数的自变量是区分大小写的。字符串文字自变量必须是类型为 XML 的限定列名。此函数允许抽取整个 XML 数据列而不应用搜索条件。

db2-fn:sqlquery 函数：普通 XQuery 允许且只允许访问 XML 数据。如果只处理 XML 数据，那么这种方案非常好，但如果应用程序需要利用两种语言和数据模型的全部力量，综合访问 XML 数据和关系型数据，那么普通 XQuery 不足以胜任。SQL/XML（即嵌入了 XQuery 的 SQL）和嵌入了 SQL 的 XQuery 都可完成此任务。要使用纳入了 SQL 的 XQuery，就需要 db2-fn:sqlquery 函数。

db2-fn:sqlquery 函数采用了一个表示全查询的字符串自变量，并返回一个由全查询返回的 XML 值并置而成的 XML 序列。全查询必须指定单列结果集，而且列的数据类型必须为 XML。通过指定全查询，将允许使用 SQL 的功能向 XQuery 提供 XML 数据。该函数支持使用参数向 SQL 语句传递值。

2．XQuery 的优缺点

XQuery 的优点如下：

（1）对于仅涉及 XML、不需要（或不希望）使用 SQL 或关系结构的应用程序来说，XQuery 很适用。

（2）对于从仅包含 XML 的数据库到 DB2 V9 的移植来说，XQuery 很适用。现有 XQuery 很可能只需要作出很小的修改即可在 DB2 中运行。例如，DB2 中 XQuery 的数据输入来自 db2-fn:xmlcolumn() 函数，而其他数据库可能成为 collection()。在这种情况下，只需重新对函数进行命名即可。

（3）如果查询结果需要嵌入新构造的 XML 文档（此文档与数据库中其他 XML 文档不同）中（并以这个新文档的形式返回），那么 XQuery 有优势。

（4）XQuery 非常适合表示两个 XML 文档之间的连接以及联合 XML 值。

XQuery 的缺点如下：

（1）使用普通 XQuery，用户无法利用 DB2 Net Search Extender（NSE）所提供的全文本搜索功能。要进行全文本搜索，则需要包含 SQL（在安装 DB2 时，默认不安装 DB2 Net Search Extender 插件，DB2 Net Search Extender 插件需要另外安装）。

（2）普通 XQuery 不允许调用 SQL 用户定义的函数（UDF）或者用 C 语言或 Java 编写的外部 UDF。

（3）目前，DB2 不提供调用带参数标记的独立 XQuery 的方法。如果要向 XQuery 传递参数，则必须使用 SQL/XMl 将一个参数标记（“?”）转换为指定变量。

3. 嵌入了 SQL 的 XQuery 的优缺点

嵌入了 SQL 的 XQuery 的优点如下：

（1）如果希望以关系列上的条件为依据，只处理 XML 文档的子集，那么嵌入了 SQL 的 XQuery 非常适用。特别是可以使用关系型谓词来约束特定 XQuery 的输入。为此，DB2 提供了另外一个输入函数 db2-fn:sqlquery，以便在 XQuery 中调用 SQL 查询。此函数接受 SQL SELECT 语句，并返回 XML 列作为输出。

（2）嵌入了 SQL 的 XQuery 允许利用全文本搜索，可在嵌入式 SQL 语句的 where 子句中使用文本搜索函数 contains。

（3）嵌入了 SQL 的 XQuery 对于那些需要整合关系型数据和 XML 数据的应用程序非常有用。嵌入了 SQL 的 XQuery 对于将关系型数据提供给 XQuery 来说非常有用。可以通过组合方式查询 XML 数据与关系型数据，这同样适用于 SQL/XML。但可能会发现，在 SQL/XML 中利用 XMLEXISTS 函数连接 XML 值和关系值更方便。

（4）嵌入了 SQL 的 XQuery 比较出色，因为嵌入式 SQL 语句可包含对用户定义的函数（UDF）的调用。UDF 在许多 IT 组织内都得到了广泛应用，用于封装关键业务逻辑和简化应用程序开发需求。这非常重要，因为普通 XQuery 不能调用 UDF。

嵌入了 SQL 的 XQuery 的缺点如下：

（1）尽管 db2-fn:sqlquery 允许在 XQuery 内嵌入 SQL 语句，但目前还不能在嵌入式 SQL 语句中使用参数标记。

（2）目前，db-2fn:sqlquery 不允许从 XQuery 向 SQL 中传递参数。因此在结合 XML 数据和关系型数据方面，SQL/XML 更强大。例如，嵌入了 SQL 的 XQuery 无法像 SQL/XML 那样用于表示关系和 XML 列之间的连接。

17.2.5　SQL/XML

SQL/XML 是 SQL 语言标准的一部分，它定义了一种新的 XML 数据类型以及多种查询、构造、验证和转换 XML 数据的函数。早在 DB2 V8 中就加入了许多 SQL/XML 发布函数，使用户能够使用 XMLELEMENT、XMLATTRIBUTE、XMLFOREST、XMLAGG 和其他一些函数将关系型数据构造为 XML。在 DB2 V9 添加了新的 SQL/XML 查询函数，包括 XMLQUERY、XMLTABLE 函数以及 XMLEXISTS 谓词。这些函数允许用户在 SQL 语句中嵌入 XQuery 或简单的 XPath 表达式。

1. XMLQUERY 函数

XMLQUERY 是一个标量函数，它将 XQuery 表达式作为自变量，返回的是一个 XML 序列。该函数有一些可选函数，可用来将 SQL 值作为 XQuery 变量传递给 XQuery 表达式。可以在 SQL 查询的上下文中进一步处理由 XMLQUERY 返回的 XML 值。

XMLQUERY 函数的语法如下：

```
XMLQUERY([xmlnamespaces-declaration]
Xquery-expression-constant
```

```
[PASSING [BY REF]xquery-argument]
[RETURNING SEQUENCE[BY REF]]
[empty on empty])
```

其中：

```
xquery-argument::=xquery-variable-expression AS identifier[BY REF]
```

参数说明如下。

- ☑ XMLQUERY 函数的模式名为 SYSIBM。
- ☑ XMLQUERY 的结果的数据类型是 XML，不允许为空。
- ☑ Xmlnamespaces-declaration：指定一个或多个 XML 空间声明。
- ☑ Xquery-expression-constant：指定一个 SQL 字符串常量，它是一个 XQuery 查询表达式返回的结果。在被作为 XQuery 语句解析之前，常量字符串转换为 UTF-8 代码。
- ☑ PASSING：指定 xquery-expression-constant 表达式的输入参数。默认情况下，XMLQUERY 函数调用 xquery-argument 中的每个列名并且隐含地以列名作为变量名传递给 xquery-expression-constant 表达式。如果 xquery-argument 中的一个 identifier 匹配 xquery-expression-constant 表达式中的一个列名，那么 xquery-argument 就传递给 xquery-expression-constant 表达式，覆盖隐式的列名。
- ☑ BY REF：指定 XML 返回值的数据类型数据作为参照，BY REF 是默认的。
- ☑ Xquery-varable-expression：指定一个 SQL 表达式，在执行查询的过程中，这个 SQL 表达式的值可以被 XQuery 表达式 xquery-expression-constant 引用。
- ☑ Identifier：xquery-variable-expression 产生的值将作为一个 XQuery 变量被传递到 xquery-expression-constant，这个变量的名称是 identifier。
- ☑ RETURNING SEQUENCE：指定 XMLQUERY 表达式返回一个序列。
- ☑ EMPTY ON EMPTY：指定处理 XQuery 表达式产生的空序列作为空序列返回。

2. XMLTABLE 函数

XMLTABLE 是一个表函数，它使用 XQuery 表达式从 XML 数据生成 SQL 表，这样使得 SQL 可以进一步处理该 SQL 表。

XMLTABLE 函数的语法为：

```
XMLTABLE([xmlnamespaces-declaration]
Row-xquery-expression-constant
[PASSING[BY REF]row-xquery-argument]
[COLUMNS xml-table-regular-column-definition])
```

其中：

```
row-xquery-argument::=xquery-variable-expression AS identifier[BY REF]
xml-table-regular-column-definition::=
column-name data-type[BY REF][WITH]DEFAULT[default-values]
[PATH column-xquery-expression-constant]
```

参数说明如下。

- ☑ XMLQUERY 函数的模式名为 SYSIBM。
- ☑ XMLTABLE 函数将 XQuery 表达式的值作为一个结果表（虚拟表）返回，可能使用指定的输入参数作为 XQuery 的变量。XQuery 表达式的结果序列的每个序列项代表结果表的一行。
- ☑ Row-xquery-expression-constant：是一个返回 SQL 字符串常量的 XQuery 表达式。字符串常量直接转换到 UTF-8，而不转换到数据库代码页。row-xquery-expression-constant 的值不能是空字符串或者全是空格的字符串。
- ☑ PASSING：指定 row-xquery-expression-constant 表达式的输入参数。默认情况下，XMLQUERY 函数调用范围内的每个列名，隐含地以列名作为变量名传递给 XQuery 表达式。如果 row-xquery-argument 中的一个 identifier 匹配 row-xquery-expression-constant 表达式，那么覆盖隐式的列名。
- ☑ COLUMNS xml-table-regular-column-definition：指定结果表的输出列，包含列名、数据类型、XML 传参机制和 XQuery 表达式，其中 XQuery 表达式用于提取结果表的行中的序列项的值。
- ☑ Column-name：指定结果表中的列名。如果要引用结果列，必须指定 column-name。列名不能有前缀，一个表中不允许出现重名的列。
- ☑ Data-type：指定结果表中列的数据类型。
- ☑ [WITH] DEFAULT [default-values]：指定结果表中列的默认值，当 column-xquery-expression-constant 表达式返回空序列时，默认值就指定给该列。
- ☑ PATH column-xquery-expression-constant：是一个返回 SQL 字符串常量的 XQuery 表达式。字符串常量直接转换到 UTF-8，而不转换到数据库代码页。可隐含地用 XMLCAST 函数将 column-xquery-expression-constant 表达式的返回值转换为 COLUMNS 子句中的列的数据类型，且被作为 COLUMNS 子句中的列的值，column-xquery-expression-constant 表达式是 row-xquery-expression-constant 表达式中的项。column-xquery-expression-constant 的值不能是空字符串或者全是空格的字符串。

3．XMLEXISTS 谓词

XMLEXISTS 是一个 SQL 谓词，它确定 XQuery 表达式是否返回由一个或多个项组成的序列（而不是一个空序列）。

XMLEXISTS 谓词的语法为：

```
XMLEXISTS(xquery[xmlnamespaces-declaration]
expression-constant
[PASSING[BY REF]xquery-argument])
```

其中：

```
xquery-argument::=
xquery-variable-expression AS identifier[BY REF]
```

4．SQL/XML 的优缺点

SQL/XML 的优点如下：

（1）如果当前有一个 SQL 应用程序，需要在各处逐步添加一些 XML 功能，那么 SQL/XML 具有一定的优势。

（2）如果用户是 SQL 的忠实爱好者，并且希望以 SQL 为首选语言——因为它是数据库用户和开发团队最熟悉的语言，那么 SQL/XML 有优势。

（3）如果查询需要同时从关系型列和 XML 列中返回数据，那么 SQL/XML 有优势。

（4）如果查询需要全文本搜索条件，那么 SQL/XML 有优势。

（5）如果希望将结果作为集合返回，且将丢失的 XML 元素表示为空，那么 SQL/XML 有优势。

（6）如果希望使用参数标记，那么 SQL/XML 有优势，因为 DB2 V9 XQuery 不支持外部参数。XMLQUERY、XMLTABLE 和 XMLEXISTS 中的传递机制允许将一个 SQL 参数标记作为变量（例如$x）传递到嵌入式 XQuery 表达式中。

SQL/XML 的缺点如下：

（1）对于将 XML 文档转换成另一个 XML 文档，SQL/XML 并非总是最优选择。在只需要处理 XML 数据时，单独用 XQuery 通常更合适，也更直观。

（2）在 SQL/XML 中表示两个 XML 列（或者更常见的情况，两个 XML 值之间的连接）非常不方便，用 XQuery 通常更高效、更直观。

17.3 更新 XML 数据

本节将讲述使用 SQL 语句更新和删除 XML 数据，读者可以自行向 XML 中插入数据，另外还介绍变换表达式的基本语法。

17.3.1 使用命令更新 XML 数据

【例 17.11】 更新姓名为“马芳”的学生的“联系方式”列中的数据（**实例位置：光盘\TM\sl\17\3**）。实现步骤如下。

（1）连接 MRKJ 数据库：

```
CONNECT TO MRKJ USER MR USING 1;
```

（2）更新姓名为“马芳”的学生的“联系方式”列中的数据：

```
update mr.学生表
set 联系方式=
'<contact name ="马芳">
<email>MF112@165.com</email>
<phone type ="mobile">186××××××××</phone>
<phone type ="home">668888</phone>
<native_place country="中国">
<province>山东</province>
<city>青岛</city>
```

```
</native_place>
</contact>'
where 学号='14001';
```

本例运行结果如图 17.6 所示，单击“...”按钮便可查看信息。



图 17.6　更改结果

【例 17.12】　删除姓名为“马芳”的学生的“联系方式”列中的数据（**实例位置：光盘\TM\sl\17\4**）。

实现步骤如下。

（1）连接 MRKJ 数据库：

```
CONNECT TO MRKJ USER MR USING 1;
```

（2）删除姓名为“马芳”的学生的“联系方式”列中的数据：

```
update mr.学生表
set 联系方式=NULL
where 姓名='马芳';
--删除姓名为"马芳"的学生的"联系方式"字段
```

使用 SQL 语句删除和插入 XML 数据与使用 SQL 语句更新 XML 数据类似。

17.3.2　变换表达式

变换表达式会创建一个或多个节点的副本。变换表达式 modify 子句中的更新表达式会更改复制的节点。Return 子句中的表达式会指定编号表达式的结果。

变换表达式的语法为：

```
Transfrom <copy_clause> <modify_clause> <return_clause>
```

其中：

```
copy_clause ::=
    copy $VarableName :=CopySourceExpression
```

```
modify_clause ::=
    modify ModifyExpression
return_clause ::=
    return ReturnExpression
ModifyExpression ::=
    DeleteExpression | InsertExpression | RenameExpression | ReplaceExpression
DeleteExpression ::=
    Do delete TargetExpression1
InsertExpression ::=
    Do insert SourceExpression1
    <before | after | as first into | as last into | into > TargetExpression2
RenameExpression ::=
    Do rename TargetExpression3 as NewNameExpression
ReplaceExpression ::=
    Do replace [value of] TargetExpression4 with SourceExpression2
```

参数说明如下。

- ☑ CopySourceExpression：该表达式必须返回节点及其后代、节点数和节点层次结构的根节点。*Copy 子句创建的节点具有新的节点标识并且属于无类型节点。例如，如果复制类型为 xs:decimal 的元素节点的副本，那么复制节点有自己的节点标识，并且其类型为 xs:anyType。*
- ☑ ModifyExpression：更新表达式或空序列，将对更新表达式求值，并且会对 copy 子句创建的节点应用生成的更新。如果更新表达式的目标节点并非包含变换表达式 copy 子句创建的节点，那么 DB2 XQuery 会返回错误。
- ☑ ReturnExpression：该表达式可访问通过在 modify 子句中更新表达式而更改或创建的节点。ReturnExpression 可返回复制节点、原始节点和构造节点的任何组合。如果 ReturnExpression 是包括无类型节点（如新节点或复制节点）的 XML 文档，那么可使用 SQL/XML 函数 XMLVALIDATE 来对 XML 文档中的节点指定特定类型的信息。
- ☑ Before after as first into as last into into：用于指定 SourceExpression1 节点的关键字将成为 TargetExpression2 节点的前导同代节点/后续同代节点/第一批子代/最后一批子代/子代。如果是“before”或“after”，且插入序列包含先出现在序列中的树形节点，那么这些属性节点会成为目标节点的属性。
- ☑ Value of ：指定在替换 TargetExpression4 节点值的同时保留节点标识的关键字。如果指定 value of 关键字，那么 SourceExpression2 的结果是单个文本节点或空序列。如果未指定 value of 关键字，那么 SourceExpression2 的结果必须是节点序列。
- ☑ SourceExpression2 序列必须由下列节点类型组成：如果 TargetExpression4 节点是属性节点，那么替换序列 SourceExpression2 必须由零个或多个属性节点组成；如果 TargetExpression 节点是元素、文本、注释或处理指令节点，那么替换序列 SourceExpression2 必须由零个或多个元素、文本、注释或处理指令节点的某些组合组成。
- ☑ 下列更新是在指定 value of 关键字后生成的：如果 TargetExpression4 节点是元素节点，那么 TargetExpression4 节点的现有子代将替换为 SourceExpression 返回的文本节点。如果 SourceExpression2 返回空序列，那么 TargetExpression4 节点的子代属性将为空。如果 TargetExpression4 节点包含属性节点，那么它们不受影响。如果 TargetExpression4 节点并非元

素节点，那么 TargetExpression4 节点的字符串值将替换为 SourceExpression2 返回的文本节点的字符串值。如果 SourceExpression2 未返回文本节点，那么 TargetExpression4 节点的字符串值将替换为零长度字符串。

☑ 下列更新是在未指定 value of 关键字的情况下生成的：

- SourceExpression2 节点会替换 TargetExpression4 节点。SourceExpression2 节点占据了节点层次结构中原来由 TargetExpression4 节点占据的位置。
- TargetExpression4 节点及其所有属性和后代将与节点序列分离。

17.4　小　　结

本章主要对 XQuery 技术进行详细讲解。在具体讲解过程中，首先介绍了 XQuery 基础，让读者对 XQuery 技术查询的组成部分、基本规定、XML 名称空间、Qname、序言和表达式有一个清晰的了解；然后讲解了查询 XML 数据及更新 XML 数据等内容。通过本章的学习，读者应该熟练掌握 XQuery 技术的使用，并能够使用 XQuery 技术解决 DB2 数据库开发中遇到的问题。

17.5　实践与练习

（1）XQuery 技术的基本规定有哪些？（**答案位置：光盘\TM\sl\17\5**）

（2）请列举出任意两种查询 XML 的方法。（**答案位置：光盘\TM\sl\17\6**）

第18章

数据移动

（视频讲解：68分钟）

数据移动也许是数据库运维中最常见的工作，将一些数据从A数据库输入B数据库，如何才能简单有效地完成这样的工作，是很多DBA的一大挑战。DB2提供了多种工具可以选择，例如import、load。本章将从数据移动、文件格式、数据的导出、数据的导入、数据的装入、复制表等多个方面对DB2中数据移动进行详细讲解。

通过阅读本章，您可以：

- 了解数据移动的概念
- 了解文件格式
- 掌握数据的导出
- 掌握数据的导入
- 掌握数据的装入
- 掌握如何进行表的复制
- 熟悉数据移动时的一些常见问题

18.1　数据移动概述

数据移动也叫数据迁移，是 DBA 重要的工作内容之一。广义的数据移动，既包括平台间数据库移动也包括同一数据库不同表间的数据迁移，还包括数据表与文件之间的数据导入/导出。针对不同场景，DBA 要在众多的方案中选择最优的解决方案。

同构平台间的数据迁移，如从生产库系统到测试库，最简单的迁移方法是通过对生产库做备份，然后在测试机上进行数据库恢复。

异构平台间的数据迁移，如从 Windows 平台到 Linux 系统的数据迁移，这种场景不能采用数据库备份恢复，只能先用 db2look 导出表结构，并将表数据导出来，然后再导入到目标库。

灾难或故障情况下的数据挽回，DB2 提供了 db2dart 工具，可以在实例都无法启动的情况下将数据导出。该工具一般用于没有进行数据库备份，但发生了日志故障或其他操作故障，而没有其他机制挽回数据的场景。

DB2 中所谓的数据移动，包括以下内容：

- ☑ 数据的导入（Import）。
- ☑ 数据的导出（Export）。
- ☑ 数据的载入（Load）。
- ☑ db2move：用来批量移动多个表格。
- ☑ ADMIN_COPY_SCHEMA 过程：允许生成单个模式中所有对象的副本，然后在新模式中重新创建这些对象。可以在数据库中使用数据或不使用数据执行此复制操作。
- ☑ 带有 REDIRECT 选项和 GENERATE SCRIPT 选项的 Restore 实用程序：使用现有备份映象中的脚本将整个数据库从一个系统复制到另一个系统中。
- ☑ Db2relocatedb：重新分配数据库命令。
- ☑ Db2inidb：分割镜像。
- ☑ IBM Migration Toolkit（MTK）：是一个转换工具，可用来将源数据库关系系统（例如，Informix Dynamic Server、Informix Extended Parallel Server（XPS）、Microsoft SQL Server、Oracle 和 Sybase Enterprise）中的数据以及查询的过程语言迁移到 DB2 数据库产品中。MTK 可运行于 AIX、Linux、Solaris 和 Windows 操作系统上。它只支持“英语”。

导入和装入都是利用 DB2 的相关命令把某种格式的文件中的数据保存到数据库中的表中。导出是指把 DB2 数据库的表中的数据保存到某种格式的文件当中。

数据移动的作用：如果要在不同的数据库管理系统之间转移数据，数据移动通常是最实用的一种方法，因为任何一种数据库管理系统都支持常用的几种文件格式，通过这个通用的接口，就很容易实现不同系统间数据的转移。

18.2　文 件 格 式

首先介绍 DB2 支持的文件格式：DEL、ASC、PC/IXF 和 WSF 格式，其中 DEL 和 ASC 格式是文

本格式，PC/IXF 格式是 IBM 特有的二进制格式，WSF 格式主要用于和 Lotus-1-2-3 进行数据导入/导出，新版本中将不再支持。

18.2.1 DEL 格式

定界 ASCII 格式（DEL）是 DB2 用于数据交换的最常用格式。这种格式包含 ASCII 数据，使用字符分隔符分隔列值，分隔符用来标识数据元素的开始和结束，最主要的分隔符有以下几种。

☑ 字符分隔符：界定字符字段的起始。在默认情况下，用双引号（“”）作为字符分隔符。

☑ 列分隔符：界定列的结束。默认情况下用逗号（,），作为列分隔符。

☑ 行分隔符：用来标识一行或一个记录的结束。默认情况下用换行符作为行分隔符。

```
"101","计算机应用技术",1,90,6
"102","学前教育",2,78,5
```

18.2.2 ASC 格式

非定界 ASCII 格式（ASC）（对于 Import 和 Load 实用程序来说称为 ASC）有两种变体：定长 ASC 和变长 ASC。对于定长 ASC 来说，所有记录都是定长的。对于变长 ASC 来说，记录由行定界符（始终是换行符）定界。在非定界 ASCII 中，术语非定界表示列值未由定界符分割。在导入或装入 ASC 数据时，如果指定了 reclen 文件类修饰符，那么表示数据文件是定长 ASC 文件。如果未指定该修饰符，那么表示该数据文件是变长 ASC 文件。

例如：

```
"101","计算机应用技术",1,90,6
"102","学前教育",2,78,5
```

18.2.3 PC/IXF 格式

PC/IXF 格式是 IBM 特定的二进制格式，也是集成交换格式（Integration Exchange Format，IXF）数据交换体系结构的改编版本，由一些列可变长度的记录构成，包括头记录、表记录、表中每列的列描述符记录以及表中每行的一条或多条数据记录。PC/IXF 文件记录由包含了字符数据的字段组成。

18.2.4 WSF 格式

WSF 格式是工作表格式，用于 Lotus 1-2-3 和 Symphony 之类的产品的数据交换。Load 实用程序不支持此文件格式。

18.2.5 CURSOR格式

游标(Cursor)是非常重要的一个概念，它提供了一种对从表中检索出的数据进行操作的灵活手段。可以把游标想象为一个指针，刚开始指向结果集中的第一条数据，当第一条数据读取完成后，游标会自动跳转到下一条数据。

如果在两张表之间进行数据迁移，最简单的方法是先将数据从一张表导出来，存到一个文件中，然后将这个文件的数据导入到另外一张表。如果使用游标，数据不需落地，效率比较高，因此CURSOR格式比较适用于表间数据迁移。

18.3 数据的导出

Export（数据的导出）实用程序会使用SQLSELECT语句或XQuery语句抽取数据，并将该信息放到文件中。用户可使用输出文件移动数据，以便将来执行导入或装入操作，或者将数据用于分析。

执行Export需要SYSADM、SYSCTRL或者SYSMAINT权限，或者是对输出的表格或视图有CONTROL权限。

18.3.1 使用EXPORT导出数据

Export是一个相对简单而且具有操作灵活性特点的数据移动实用程序。可通过下列方法将它激活：通过控制中心，在“命令行处理器”或“命令编辑器”中发出EXPORT命令，调用ADMIN_CMD存储过程或通过应用程序调用db2Export API。

【例18.1】 导出“学生表”的所有数据。

通过控制中心导出数据的实现步骤如下：

(1)创建用于存放输出文件和消息文件的路径(例如，D:\mrkj\datamovement)，创建用于存放LOB文件的路径（例如，D:\mrkj\datamovement\lobpath），创建用于存放XML文件的路径（例如，d:\mrkj\datamovement\xmlpath)。

(2)在“控制中心”对象视图中，选择MRKJ数据库的“表”文件夹，在内容窗格中找到“学生表”，在该表单击鼠标右键，在弹出的快捷菜单中选择“导出”命令，如图18.1所示。

(3)在弹出的“导出表”对话框的“目标”选项卡上，在“输出文件”后面的文本框中输入“输出文件”路径和名称“D:\mrkj\datamovement\xsb_data.csv”(该路径必须已经存在，但该文件可以不存在，可通过单击“输出文件”后面的文本框后面的“...”按钮及单击该按钮后弹出的对话框来辅助填写文件路径和名称)，在“消息文件”后面的文本框中输入“消息文件”的路径和名称“D:\mrkj\datamovement\xsb_export_msg.txt”(该路径必须已经存在，但该文件可以不存在，也可通过单击“消息文件”文本框后面的“...”按钮以及单击该按钮后弹出的对话框来辅助填写文件路径和名称)。可以修

改 SELECT 语句（可以单击“SQL 助手”按钮修改 SELECT 语句），这里选择默认值。在“文件格式”栏中选中“定界”单选按钮（即 DEL 文件格式），如图 18.2 所示，单击“选项”按钮。

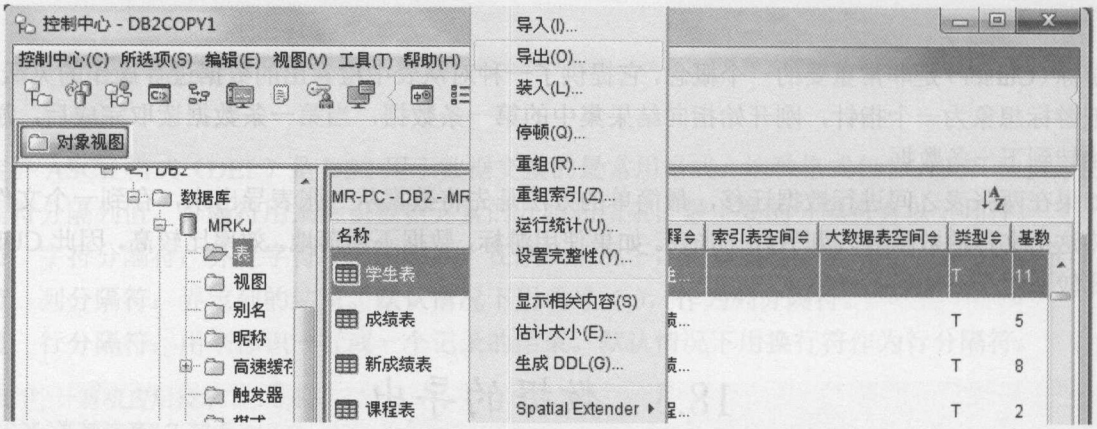


图 18.1　选择“导出”命令

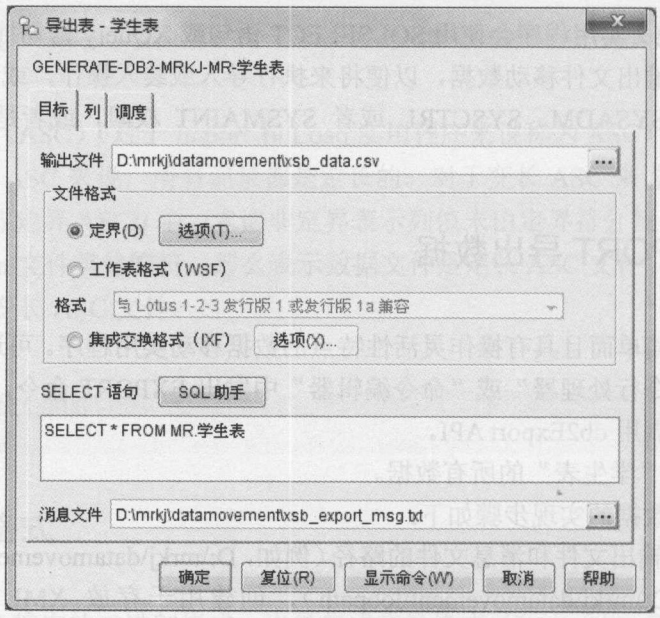


图 18.2　“导出表-学生表”对话框

注意

（1）.csv 是一种特殊的文本文件格式，它可以用 Excel 工具打开，存储类似于 DEL 或 ASC 格式的文件，但列定界符只限于逗号和空格。Windows 操作系统默认的文件类型没有.csv，所以上面的文件的扩展名.csv 需要手工输入。

（2）在使用 Export 语句之前，必须创建用于存放输出文件和消息文件的目录。

（3）这里输出文件名称的扩展名和消息文件名称的扩展名可以任意写，系统都不会报错，但本质上它们都是文本文件，所以最好都写成文本文件扩展名，例如.csv 或.txt。

（4）在打开的“导出定界的选项”对话框中，可以根据需要定制 DEL 文件的各种属性。选中“使用 ISO 日期格式”复选框，在 XML 栏中选中“将验证信息包含在 XML 数据说明符（XDS）中”复选框，然后再单击“确定”按钮返回到“导出表”对话框，如图 18.3 所示。

（5）选择“导出表-学生表”对话框中的“列”选项卡，在“指定用于放置大对象（LOB）数据的一个或多个路径”下面的文本框中输入“D:\mrkj\datamovement\lobpath”（该路径必须已经存在，可通过单击该文本框后面的“...”按钮选择），在“指定要用于大对象（LOB）数据的基本文件名”下面的文本框中输入“lob_data.del”，选中“将每个大对象（LOB）放置在一个单独的文件中”复选框，在“指定用于放置 XML 数据的一个或多个路径”下面的文本框中输入“D:\mrkj\datamovement\ xmlpath”（该路径必须已经存在，可通过单击该文本框后面的“...”按钮选择），在“指定要用于 XML 数据的基本文件名”下面的文本框中输入“xml_data.del”，选中“将每个 XML 文档放置在一个单独的文件中”复选框，可以单击“显示命令”按钮查看自动生成的 SQL 命令。单击“确定”按钮完成“学生表”数据的导出，如图 18.4 和图 18.5 所示。

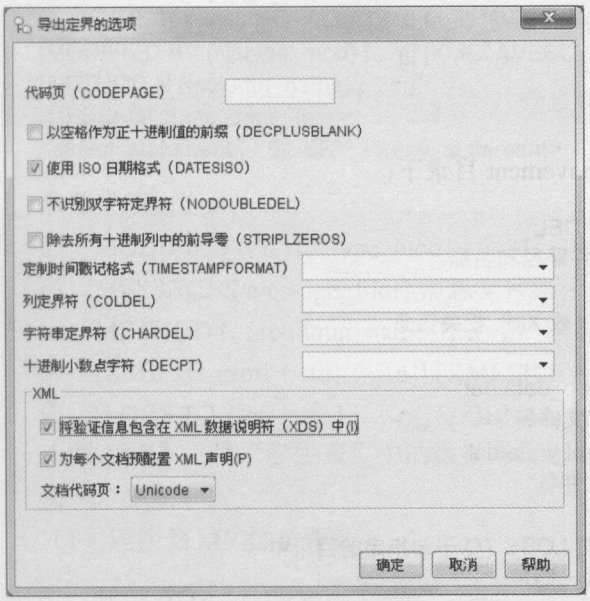


图 18.3　导出表的导出定界选项

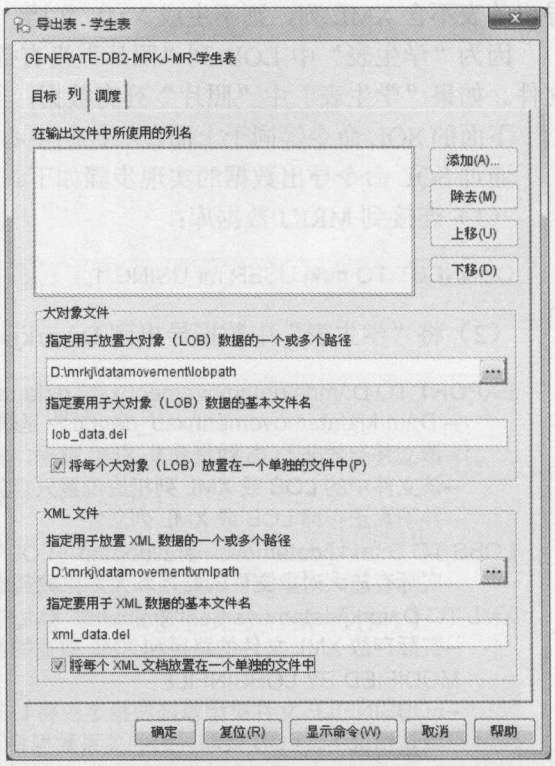


图 18.4　导出数据的列名和 LOB 与 XML 文件设置

执行结果如下。

前面的导出表操作完成后可以发现“D:\mrkj\datamovement”路径下已经出现了两个文件：

（1）xsb_data.csv——包含“学生表”中所有记录（其中 XML 列“联系方式”仅有 XML 数据说明符（XSD）和验证模式 MR.CONTACT_SCHEMA 的信息）。其中列分隔符为默认的逗号。可用写字板、记事本等文本工具或 Excel 工具打开此文件。“学生表”中的每条记录在 Excel 中都显示为一行，每个数据在 Excel 中都显示在一个单元格中。

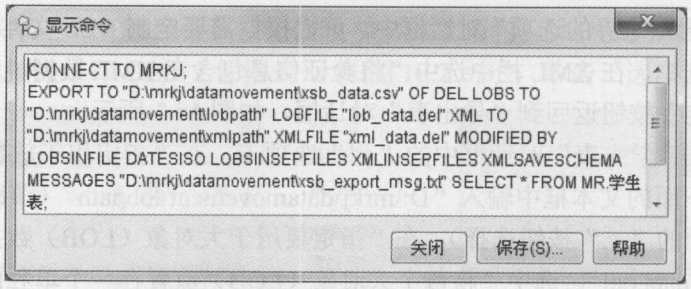


图 18.5　查看自动生成的 SQL 命令

（2）xsb_export_msg.txt——记录导出文件的过程。可用文本打开工具打开此文件。在“学生表”中“联系方式”这一列的值为空，否则在“D:\mrkj\datamovement\xmlpath”路径下出现几个 XML 文件：文件名类似于“xml_data.del.017.xml”。该文件记录“学生表”中 XML 列“联系方式”的数据。可用记事本、Internet Explore 或 XML 工具打开此文件，但是不能用写字板打开，否则会显示乱码。如果被导出的表不含 XML 列，则不生成 XML 文档，如果含有多个 XML 列，则生成多个 XML 文档。

因为“学生表”中 LOB 列“照片”没有数据，所以“D:\mrkj\datamovement\lobpath”路径下没有文件。如果“学生表”中“照片”列有数据，则每个 LOB 文件在该目录下生成一个输出文件。

下面的 SQL 命令等同于上面在“控制中心”导出“学生表”中数据的操作。

通过 SQL 命令导出数据的实现步骤如下。

（1）连接到 MRKJ 数据库：

```
CONNECT TO mrkj USER mr USING 1;
```

（2）将“学生表”中数据导出到 D:\mrkj\datamovement 目录下：

```
EXPORT TO D:\mrkj\datamovement\xsb_data.csv OF DEL
    -- D:\mrkj\datamovement\xsb_data.csv 是输出文件路径和名称
    --该文件存放非 LOB 数据和非 XML 数据
    --该文件中的 LOB 或 XML 列相应位置只存放 LOB 或 XML 相关信息
    --并不真正存储 LOB 或 XML 列文件
LOBS TO D:\mrkj\datamovement\lobpath   LOBFILE lob_data.del
    --实际存放大对象文件的路径和大对象数据的基本文件名
XML TO D:\mrkj\datamovement\xmlpath   XMLFILE xml_data.del
    --实际存放 XML 文件的路径和 XML 数据的基本文件名
    MODIFIED BY LOBSINFILE
    --LOBSINFILE 文件类型修饰符指定应将 LOB 写至 LOBS TO 子句指定的目录中
    --默认情况下，LOB 值与导出的关系数据将写至同一路径
    DATESISO
    --如果有日期数据则使用 ISO 日期格式
    LOBSINSEPFILES
    --将每个大对象（LOB）放置在一个单独的文件中
    XMLINSEPFILES
    --将每个 XML 文件放置在一个单独的文件中
    XMLSAVESCHEMA
    --如果含有 XML 数据，则将验证信息包含在 XML 数据说明符（XSD）中
MESSAGES D:\mrkj\datamovement\xsb_import_msg.txt
    --消息文件路径和名称，所有的导出信息都保存在该文件中
```

```
    --（无论是成功、警告还是失败信息）
    --这样，管理员可以通过观察信息文件找到问题所在
SELECT * FROM mr.学生表;
    --SELECT 语句的查询结果就是即将被输出的信息
```

用 Office Excel 工具导出（此种方式不支持 XML 和 BLOB 数据）：

先通过 Windows 控制面板配置 MRKJ 数据源，然后通过 Excel 工具的 ODBC 数据源接口将“学生表”数据导入 Excel 中。配置 ODBC 数据源方法请查阅其他资料。要利用 Excel 工具导入外部数据，需要在 Excel 主菜单中选择“数据”菜单项，在列出的子菜单中选择“导入外部数据”→“新建数据库查询”命令。将数据导出成 Excel 中的表格以后，将该表格另存为 xsb_data.csv。这里本书不作详细介绍，请读者自己练习。

18.3.2　EXPORT 命令格式及选项

简化后的 EXPORT 命令的格式为：

```
EXPORT TO filename OF filetype
    [LOBS TO{lob_path}[,..n]] [LOBFILE {filename}[,..n]]
    [XML TO {xml_path}[,..n]] [XMLFILE {filename}[,..n]]
[MODIFIED BY {filetype_mod} [,..n]] [XMLSAVESCHEMA]
[METHOD N ({column_name} [,…n])]
[MESSAGES message_file]
<select_statement | XQUERY xquery_statement>
```

参数说明如下。

☑　MODIFIED BY {filetype_mod} [,…n]：定制格式控制符语句。

☑　MESSAGES message_file：消息文件路径和名称，所有的导出信息都保存到该文件中。

☑　METHOD N ({column_name}[,…n])：给导出数据列重命名。

☑　[XML TO {xml_path} [,…n]] [XMLFILE {filename} [,…n]]：控制 XML 文件的导出。

☑　[LOBS TO {lob_path} [,…n]] [LOBFILE {filename} [,…n]]：控制 LOB 文件的导出。

【例 18.2】　将“学生表”中的数据导出为 IXF 格式（**实例位置：光盘\TM\sl\18\1**）。

实现步骤如下。

（1）连接到 MRKJ 数据库：

```
CONNECT TO mrkj USER mr USING 1;
```

（2）将“学生表”中的数据导出为 IXF 格式：

```
EXPORT   TO D:\mrkj\datamovement\xsb_data.ixf OF IXF
MESSAGES D:\mrkj\datamovement\xsb_export_msg.txt
SELECT * FROM mr.学生表;
```

执行结果如下。

执行命令后，“D:\mrkj”路径下就多出了 xsb_data.ixf 文件，且 xsb_export_msg.txt 文件内容也增加

了几条执行情况的信息，如图 18.6 所示。

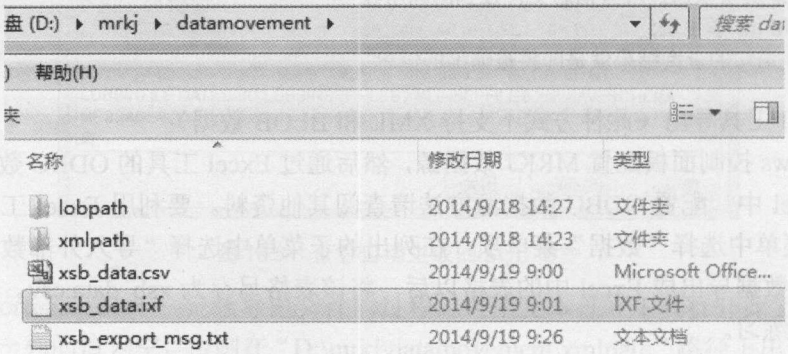


图 18.6　xsb_data.ixf 文件

下面讨论 EXPORT 命令中的各选项。

1. MODIFIED BY 子句定制格式控制符

【例 18.3】 导出“学生表”中的所有数据，导出为 DEL 格式文件（**实例位置：光盘\TM\sl\18\2**）。实现步骤如下。

（1）连接到 MRKJ 数据库：

```
CONNECT TO mrkj USER mr USING 1;
```

（2）导出“学生表”中所有数据：

```
EXPORT TO D:\mrkj\datamovement\xsbdata.csv OF DEL
MODIFIED BY COLDEL$ CHARDEL" DECPLUSBLANK DATESISO XMLSAVESCHEMA
MESSAGES D:\mrkj\datamovement\xsb_export_msg.txt
SELECT * FROM mr.学生表;
```

本例运行结果如图 18.7 所示。



图 18.7　xsbdata.csv 文件

这时可以发现，虽然导出的 xsbdata.csv 文件名的扩展名仍然是.csv 且仍然可以用 Excel 工具打开，但 Excel 工具无法识别列定界符“$”，所以“学生表”中的一行数据在 Excel 工具中都显示在一个单元格中。

在该例中，MODIFIED 子句用于定制格式控制符。

（1）COLDEL 表示列定界符，默认情况为逗号；“COLDEL$”表示以$为列定界符；如果以逗号为列定界符，则可省略 COLDEL 子句；如果是其他定界符，则不能省略。

注意

COLDEL 子句中“COLDEL”和逗号（,）必须连起来，中间不能有空格。

（2）CHARDEL 表示字符串字段用什么符号引用，默认情况下为一对双引号括起来，本示例中改为用一对单引号括起来。

（3）DECPLUSBLANK 表示对于十进制数据类型，用空格代替最前面的加号，因为默认情况下会在十进制数据前面加上正负号。

（4）DATESISO 和 XMLSAVESCHEMA 也是格式控制关键字，这在前面已经介绍过。

【例 18.4】 将“学生表”中的所有数据导出为 WSF 格式文件（**实例位置：光盘\TM\sl\18\3**）。

实现步骤如下。

（1）连接到 MRKJ 数据库：

```
CONNECT TO mrkj USER mr USING 1;
```

（2）将“学生表”中所有数据导出为 WSF 格式：

```
EXPORT TO D:\mrkj\datamovement\xsbdata.wsf OF WSF
MODIFIED BY 1
--1: 表示创建一个与 Lotus 1-2-3 Release 1 匹配的 WSF 文件，这是默认的，可以省略。
MESSAGES D:\mrkj\datamovement\xsb_export_msg.txt
SELECT 学号,姓名,性别,出生日期,专业,总分数,备注,照片 FROM mr.学生表;
```

本例运行结果如图 18.8 所示。

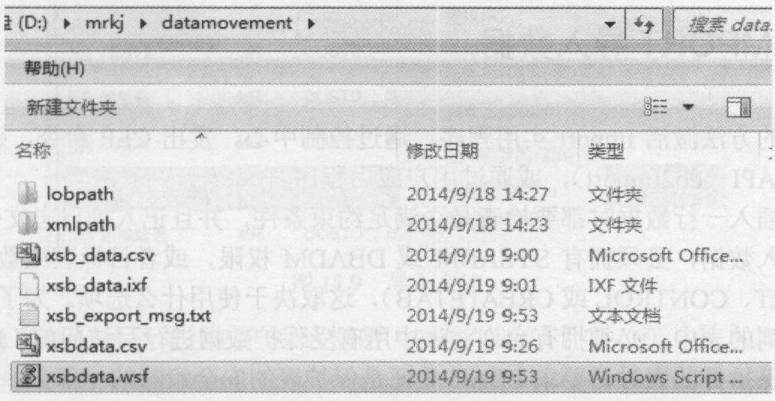


图 18.8　xsbdata.wsf 文件

执行命令后，D:\mrkj 目录下就会出现 xsb_export_msg.txt 和 xsbdata.wsf 文件。

注意

WSF 格式不支持 XML 数据。

2．给导出数据列重命名

【例 18.5】 将“学生表”的列重命名为（c1,c2,c3c,c4,c5,c6,c7,c8,c9）后，导出到 DEL 文件中（实例位置：光盘\TM\sl\18\4）。

实现步骤如下。

（1）连接到 MRKJ 数据库：

```
CONNECT TO mrkj USER mr USING 1;
```

（2）将“学生表”中的列重命名后导出到文件中：

```
EXPORT TO D:\mrkj\datamovement\xsbdata.csv OF DEL
METHOD N (c1,c2,c3,c4,c5,c6,c7,c8,c9)
MESSAGES D:\mrkj\datamovement\xsb_export_msg.txt
SELECT *  FROM mr.学生表;
```

说明

在默认情况下，导出的每一列数据以表中的对应的字段名自动命名，可以通过 METHODN 子句给每一列重新命名。

18.4 数据的导入

数据的导入（Import）实用程序会使用 SQL INSERT 语句向表、类型表或视图填充数据。如果目标表和目标视图中已包含数据，那么输入数据可替换（replace）现有数据，也可追加（insert）至现有数据。

18.4.1 使用 IMPORT 导入数据

可以通过下面的方法激活 Import 实用程序：通过控制中心，发出 CLP 命令，调用 ADMIN_CMD 存储过程，调用其 API（db2Import），或通过用户应用程序。

导入操作每次插入一行数据时都要检查是否满足约束条件，并且记入到日志文件中。

要想成功地导入数据，必须拥有 SYSADM 或 DBADM 权限，或者目标表或数据库上的底层特权（SELECT、INSERT、CONTROL 或 CREATETAB），这取决于使用什么选项。为了将数据导入到一个包含受保护的行和列的表中，必须拥有允许对表中所有受保护数据进行写访问的 LBAC 凭证。此外，将数据导入包含受保护行的表时，要求 LBAC 凭证是保护表的安全策略的一部分。

【例 18.6】 先导出“学生表”中的所有数据，然后将这些数据导入新建的表“新学生表”中（实例位置：光盘\TM\sl\18\5）。

其中，“新学生表”使用如下语句创建：

```
CONNECT TO mrkj USER mr USING 1;
CREATE TABLE mr.新学生表
```

```
(
学号 CHAR(6) NOT NULL,
姓名 CHAR(16) NOT NULL,
专业 CHAR(40),
性别 CHAR(4),
出生日期 DATE,
总分数 SMALLINT WITH DEFAULT 0,
备注 VARCHAR(400),
联系方式 XML,
照片 BLOB (1M) LOGGED NOT COMPACT,
CONSTRAINT xxsb_primary_key PRIMARY KEY(学号)
--因为向“新学生表”导入信息时将使用 INSERT_UPDATE 方法，所以此处必须定义主键
)IN customdataspace1;
```

说明

先导出“学生表”中所有的数据：用例 18.1 中的方法重写导出“学生表”中的数据。然后将已经导出的数据导入到新学生表中。

将数据导入到新学生表有两种方式可以实现，一种是使用控制中心；另一种是使用命令，接下来分别讲解这两种方法。

通过控制中心导入数据的实现步骤如下：

（1）在“控制中心”对象视图中选择 MRKJ 数据库的“表”文件夹，在内容窗格中找到“新学生表”，在该表单击鼠标右键，在弹出的快捷菜单中选择“导入”命令，如图 18.9 所示。

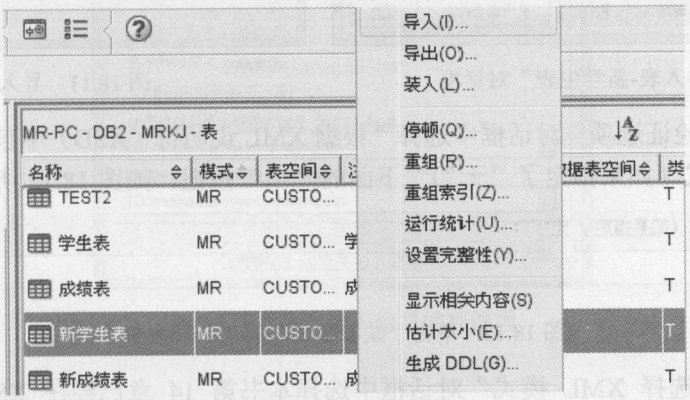


图 18.9　导入文件

（2）在打开的“导入表-新学生表”对话框的“文件”选项卡中，在“导入文件”文本框中输入导入文件的路径和名称“D:\mrkj\datamovement\xsb_data.csv”（该路径必须已经存在，但该文件可以不存在，可通过单击“导入文件”后面的“...”按钮来辅助填写文件路径和名称），在“导入方式”后面的列表框中选择“INSERT_UPDATE”，在“消息文件”文本框中输入消息文件的路径和名称“D:\mrkj\datamovement\xxsb_import_msg.txt”（该路径必须已经存在，但该文件可以不存在）。在“导入文件类型”栏中选择默认的“定界 ASCII 格式（DEL）”单选按钮，单击“DEL 选项”按钮，如图 18.10 所示。

（3）在打开的“定界选项”对话框中可以根据需要定制导入数据的各种属性。选中“使用 ISO 日

期格式”复选框，然后选中 XML 栏中的“验证入局文档”复选框，在“空格处理”后面的列表框中选择“保留空格”选项，单击 XML 栏中的“选项”按钮，如图 18.11 所示。

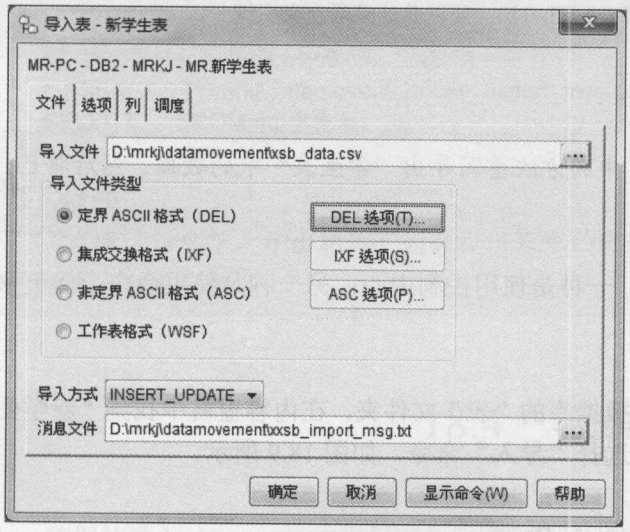


图 18.10　“导入表-新学生表”对话框

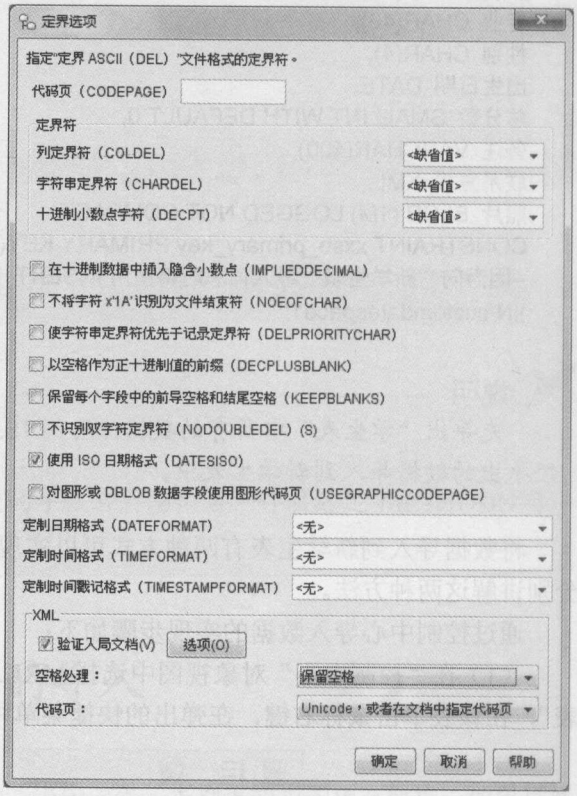


图 18.11　导入表的定界选项

（4）在打开的“验证选项”对话框中选择“根据 XML 说明符（XSD）中包含的信息来验证文档”选项，单击“默认模式（如果指定了“无”）”下面的“...”按钮，如图 18.12 所示。

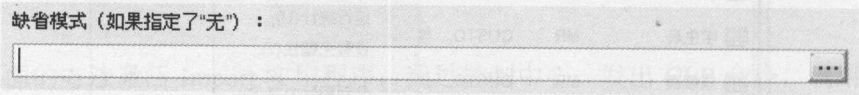


图 18.12　单击“缺省模式”的“…”按钮

（5）在打开的“选择 XML 模式”对话框中选择本书第 14 章已经在 XSR 注册的 XML 模式 MR.CONTACT_SCHEMA，如图 18.13 所示。

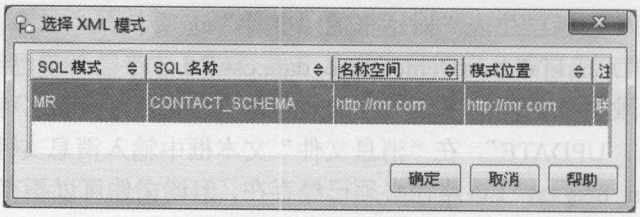


图 18.13　选择 MR.CONTACT_SCHEMA 模式

（6）在之后的对话框都单击“确定”按钮，直到回到“导入表-新学生表”对话框。

（7）选择“导入表-新学生表”对话框中的“列”选项卡，如图 18.14 所示。在“变换”栏中可以通过单击“列”按钮以及单击“列”按钮后弹出的对话框来选择要装入数据库的列号，单击“上移”或“下移”按钮可以调整目录列名的上限位置，但输入数据的列号和上下位置始终不变。因此目录列名的位置变化会导致输入数据列号与目标表“新学生表”中列名之间映射关系的改变，这里选择默认（即不调整）。选中“对空白使用默认值”复选框。在“使用这些目录来查找大对象（LOB）数据”下面的文本框中输入“D:\mrkj\datamovement\lobpath”（该目录必须已经存在，可通过单击该文本框后面的“...”按钮查找目录），在“使用这些目录来查找 XML 数据”下面的文本框中输入“D:\mrkj\datamovement\xmlpath”。可以单击“显示命令”按钮查看自动生成的 SQL 命令，单击“确定”按钮完成数据的导入。

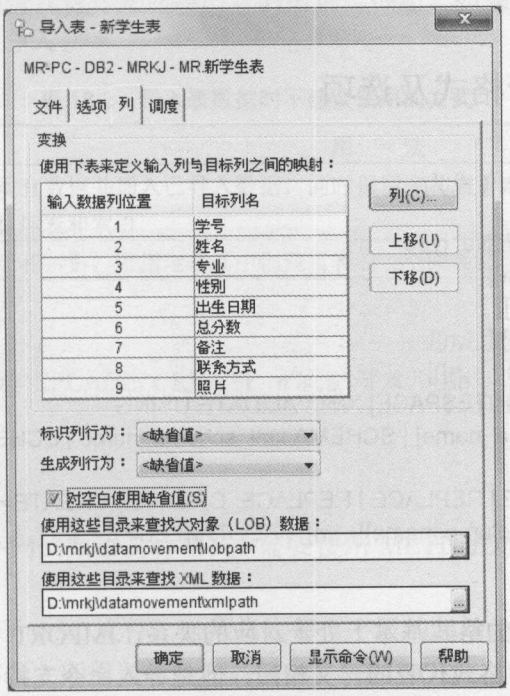


图 18.14　修改导入信息的列选项

执行结果如下：

这时打开“新学生表”可以看到该表中的数据与“学生表”中数据相同。

下面的 SQL 命令等效于上面在“控制中心”将导出的数据导入到“新学生表”的操作。

通过命令导入数据。实现步骤如下。

（1）连接到 MRKJ 数据库：

```
CONNECT TO mrkj USER mr USING 1;
```

（2）将导出的数据导入到“新学生表”中：

```
IMPORT FROM D:\mrkj\datamovement\xsb_data.csv OF DEL
LOBS FROM D:\mrkj\datamovement\lobpath
```

```
XML FROM D:\mrkj\datamovement\xmlpath
MODIFIED BY USEDEFAULTS DATESISO
METHOD P(1,2,3,4,5,6,7,8,9)
--用 P 方法指定导入的数据中的字段与目标表“新学生表”中字段的映射关系
XMLPARSE PRESERVE WHITESPACE
--解析 XML 时保留 XML 文档的元素和属性中的空格
XMLVALIDATE USING XDS DEFAULT mr.contact_schema
--用将被导入的数据中的 XDS 验证插入目标表中的 XML 文档
--默认的验证模式为已经在 XSR 中注册的 mr.contact_schema 模式
MESSAGES D:\mrkj\datamovement\xxsb_import_msg.txt
INSERT_UPDATE INTO
--用 INSERT_UPDATE 方式导入数据
mr.新学生表(学号,姓名,专业,性别,出生日期,总分数,备注,联系方式,照片);
--导入数据的目标表及目标列
```

18.4.2 IMPORT 命令格式及选项

简化后的 IMPORT 命令的格式为：

```
IMPORT FROM file_name OF file_type
[LOBS FROM {lob_path} [,…n]]
[XML FROM {xml_path} [,…n]]
[MODIFIED BY {filetype_mod} [,…n]]
[METHOD N ({column_name} [,…n])]
[XMLPARSE [PRESERVE WHITESPACE] XMLVALIDATE USING
<XDS [DEFAULT xml_schema_name] | SCHEMA xml_schema_name | SCHEMALOCATION HINTS> ]
[MESSAGES message_file]
<INSERT | INSERT_UPDATE | REPLACE | REPLACE_CREATE | CREATE>
INTO <target_table_name ({column_name}[,…n]) | ALL TABLES>
```

参数说明如下。

- ☑ 导入命令和导出命令的格式基本上处于对应的关系，IMPORT 对应 EXPORT，FROM 对应 INTO，文件名和文件格式代表的含义相同，但是导入命令支持 ASC 格式的文件，而导出的命令不支持。另外，在导出命令的最后是一个 SQL 语句，用于选择要导出的数据，而导入命令最后不是 SQL 语句，而是插入数据的方式以及目标表名称。
- ☑ 建议使用 MESSAGES 子句，以便记录错误、警告和包含有用信息的消息。
- ☑ XMLVALIDATE USING XDS[DEFAULT xml_schema_name]：根据 XML 说明符（XSD）中包含的信息来验证文档；XMLVALIDATE USING SCHEMA xml_schema_name：使用特定的 XML 模式来验证所有文档；XMLVALIDATE USING SCHEMALOCATION HINTS：根据文档的模式位置提示来验证文档。

下面讨论 IMPORT 命令中的各选项。

1. 导入方式

导入可使用 5 种方式，如表 18.1 和表 18.2 所示，它们用于确定导入数据的方法。前 3 种方式为

INSERT、INSERT_UPDATE 和 REPLACE，在目标表已存在的情况下使用。这 3 种方式都支持 IXF、WSF、ASC 和 DEL 数据格式。但是只有 INSERT 和 INSERT_UPDATE 可与昵称配合使用。

表 18.1　导入表数据时建议使用的方式

方　式	用　法
INSERT	将输入数据插入到目标表中而不更改现有数据
INSERT_UPDATE	这种方式只能用于主键的表；使用输入行的值更新具有匹配主键值的行；如果没有匹配行，那么会将已导入行插入到表中
REPLACE	删除所有

另外两种方式为 REPLACE_CREATE 和 CREATE，在目标表不存在时使用。它们只能与 PC/IXF 格式的输入文件配合使用，此格式包含要创建的表的结构化描述。如果对象表具有自身以外的任何从属，那么不能以这些方式执行导入。

表 18.2　导入表数据时不建议使用的方式

方　式	用　法
REPLACE_CREATE	删除所有现有数据并插入已导入数据，同时保留表和索引定义；如果目标表和索引不存在，那么创建目标表和索引
CREATE	创建目标表和索引，可指定在其中创建新表的表空间名称

注意

不建议使用 CREATE 和 REPLACE_CREATE 方式。请改用 db2look 实用程序。

2. METHOD 选项

IMPORT 实用程序根据不同的数据文件格式，需要使用 METHOD 选项指定数据列字段以及起始位置。

（1）METHOD L：适用于 ASC 文件，必须指定每个字段的起始位置与结束位置；每个起始位置和终止位置间用空格分开，字段之间用逗号分开。例如：METHOD L(110,12 15)。

（2）METHOD N：适用于 IXF 文件，指定字段的名称。例如：METHOD N(c1,c2,c3)。

（3）METHOD P：适用于 DEL 和 IXF 文件，指定字段的号码。例如：METHOD P(1,3,4)。

3. 文件类型修饰符

IMPORT 实用程序支持 5 种文件类型修饰符来定制导入操作。下面列出了一些修饰符。

（1）compound=x：使用非原子复合 SQL 插入数据。X 是每次将尝试的语句数量。

（2）indexschema=schema：使用索引创建期间指定的用于索引的模式。

（3）striptblanks：数据装载到可变长度字段时，截取结尾空白。

（4）lobsinfile：指明将导入 LOB 数据。IMPORT 实用程序将检查 LOBS FROM 子句，以获得输入 LOB 文件的路径。

4. 关于空值的导入

对于含有空值的数据，建议被导入的数据所在的文件采用 IXF 或 DEL 格式，导入空值非常方便，因为里面已经记录了空值的信息。但是，对于 ASC 格式文件就有一定的难度了，因为 DB2 会直接插入空格，而不是空值。为此，DB2 提供了一个子句进行控制：NULL INDICATORS 和 MODIFIED BY nullindchar 子句可用来处理空值，详情请查阅 DB2 相关资料，这里不再举例。

18.5 数据的装入

装入命令格式与导入类似，命令关键字是 LOAD，但是它的参数比导入命令要多得多，而且装入的性能要比导入高。装入与导入都是将文件中的数据移动到目标表中。在装入之前，目标表必须已经存在。装入操作不记录到日志中，所以不能使用日志文件进行前滚操作。

要将数据装入到表中，必须拥有下列其中一项权限或特权：

- ☑ SYSADM 权限。
- ☑ DBADM 权限。
- ☑ 对数据库的 LOAD 权限以及下列特权：
  - ➢ 以 INSERT 方式、TERMINATE 方式（用于终止先前的装入插入操作）或 RESTART 方式（用于重新启动先前的装入插入操作）调用 Load 实用程序时，对表的 INSERT 特权。
  - ➢ 以 REPLACE 方式、TERMINATE 方式（用于终止先前的装入替换操作）或 RESTART 方式（用于重新启动先前的装入替换操作）调用 Load 实用程序时，对表的 INSERT 和 DELETE 特权。
  - ➢ 作为装入操作使用异常表时，对异常表的 INSERT 特权。

18.5.1 装入操作的 4 个阶段

LOAD 过程由 4 个不同的阶段组成（如图 18.15 所示）。

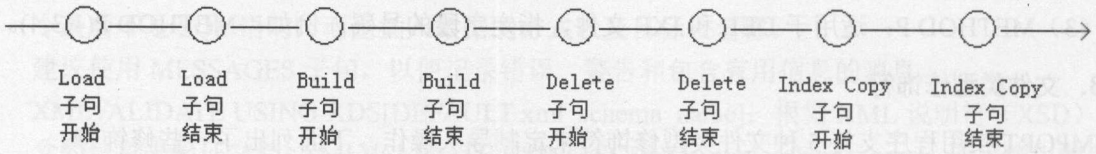


图 18.15 装入过程的 4 个阶段

（1）装入：在装入阶段，将把数据装入表中，必要时还将收集索引键和表统计信息。并且将根据 LOAD 命令中的 SAVECOUNT 参数指定的时间间隔建立保存点或一致点。将生成消息以指示在保存点成功装入的输入行数。

（2）构建：在构建阶段，将根据装入阶段收集的索引键生成索引。在装入阶段将对索引键进行排序并收集索引统计信息（如果指定了 STATISTICS USE PROFILE 选项且概要文件指示收集索引状态）。

这些统计信息与通过 RUNSTATS 命令收集的统计信息类似。

（3）删除：在删除阶段，将从表中除去导致唯一键或主键违例的行。如果指定了装入异常表，那么这些删除的行将存储在该表中。

（4）索引复制：在索引复制阶段，将索引数据从系统临时表空间复制到原始表空间。在指定了 READ ACCESS 选项的装入操作期间，只有当指定使用系统临时表空间来创建索引时，才会出现这种情况。

18.5.2　使用 LOAD 清空表数据

在不需要一个表中的数据时，需要将其清空，而做这个工作可以选择多种方法，下面分别进行介绍。

（1）删除表中数据：

```
DELETE FROM table_name;
```

这种方式可用性非常差，会消耗大量的时间和日志空间，一般不使用。除非需要有选择地删除数据时。

（2）不记日志删除表中数据：

```
ALTER TABLE table_name ACTIVATE NOT LOGGED INITIALLY;
DELETE FROM table_name;
```

这种方式可以使用，但是过于麻烦，尤其还要注意两条语句必须在同一事务中执行。

（3）修改表不记录日志方式：

```
ALTER TABLE table_name ACTIVATE NOT LOGGED WITH EMPTY TABLE;
```

这种方式是一种较好的快速清空数据表的方式。

（4）LOAD 的 REPLACE 方式：

```
LOAD FROM empfile_name OF DEL REPLACE INTO table_name;
```



注意

文件 empfile_name 为空。

这也是一种较好的快速清空数据表的方式，而且因为 LOAD 支持 RESTART 和 TERMINATE 等故障控制选项，所以方法（4）比方法（3）的过程更加可控。所以推荐使用方法（4）的方式来快速清空一张表。

【例 18.7】　用 LOAD 命令清空“新学生表”（**实例位置：光盘\TM\sl\18\6**）。



说明

首先创建一个空的文本文件“D:\mrkj\datamovement\empfile.txt”，然后用 LOAD 命令将空的文本文件导入“新学生表”，从而清空该表。

实现步骤如下。

（1）连接到 MRKJ 数据库：

```
CONNECT TO mrkj USER mr USING 1;
```

（2）用 LOAD 命令清空“新学生表”：

```
LOAD FROM D:\mekj\datamovement\empfile.txt OF DEL
REPLACE INTO mr.新学生表;
```

18.5.3 LOAD 实用程序

可以通过命令行处理器（CLP）或者 db2Load 应用程序编程接口（API）来调用 LOAD 实用程序。

【例 18.8】 将“学生表”中所有数据装入到“新学生表”中。

实现用“控制中心”的“装入向导”装入的步骤如下：

（1）在“控制中心”主菜单中选择“工具”菜单项，在列出的子菜单中选择“向导”菜单项，如图 18.16 所示。

（2）在弹出的“向导”对话框中选择“装入向导”选项，单击“确定”按钮，如图 18.17 所示。

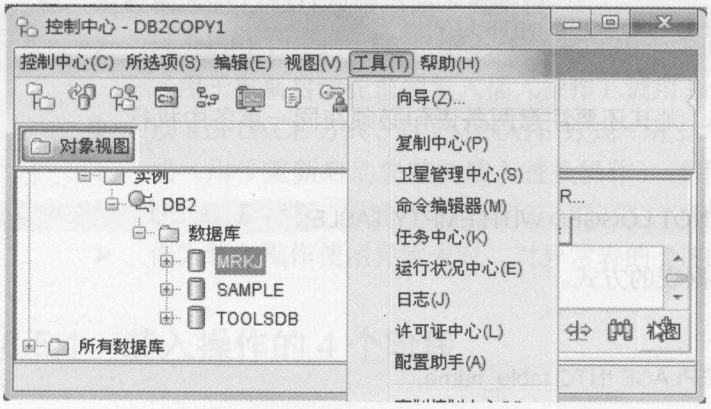


图 18.16 控制中心“工具”菜单

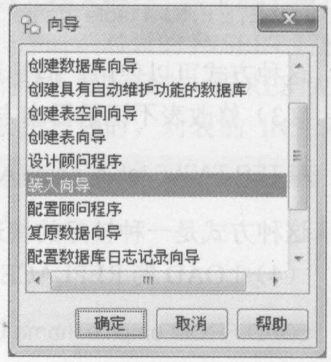


图 18.17 向导目录

（3）在弹出的“表选择”对话框的各列表框中依次选择“系统名”（例如，“MR-PC”）→“实例名”（例如，“DB2”）→“数据库名”（例如，“MRKJ”）→“表名”（例如，“mr.新学生表”），单击“确定”按钮，如图 18.18 所示。此处选择的表是即将被装入数据的目标表。

（4）在弹出的“装入向导”对话框中，如图 18.19 所示，选中“替换表数据”单选按钮，单击“下一步”按钮。

（5）指定输入、输出文件，如图 18.20 所示。在“输入文件格式”下面的列表框中选择“从游标装入”选项。手工输入（或通过单击“SQL 助手”按钮辅助输入）SELECT 语句“SELECT * FROM mr.学生表”，这表示游标指向 SELECT 语句的结果集，该结果集就是即将被装入的源数据。在“要存储进度消息的完整路径和文件名”下面的文本框中输入“D:\mrkj\datamovement\xsb_load_msg.txt”（此路径必须已经存在，但此文件当前可以不存在，可以通过单击该文本框后面的“...”按钮及单击“...”按

钮后弹出的对话框辅助输入路径和文件名）。

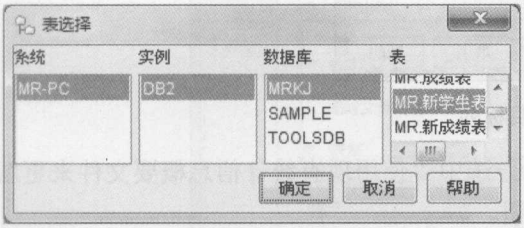


图 18.18　装入表向导

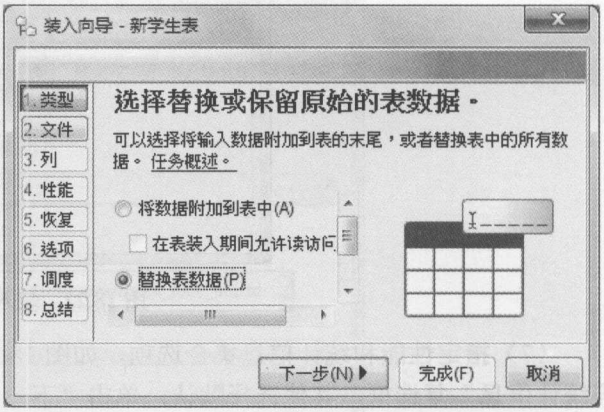


图 18.19　选择替换或保留原始的表数据

（6）定义输入列以及它们与输出列之间的映射，如图 18.21 所示。在“指定用来执行映射的方法”下面选中“基于在输入文件中找到的列名来映射列”单选按钮。可以通过单击“列”按钮以及单击“列”按钮后弹出的对话框来选择要装入数据的目标列，如图 18.22 所示。单击“上移”或“下移”按钮可以调整目标列的位置，但输入数据的列名和位置始终不变。因为目标表的列名的位置变化会导致输入数据列名与目标表“新学生表”的列名之间映射关系的改变，这里选择默认值，然后单击“下一步”按钮。

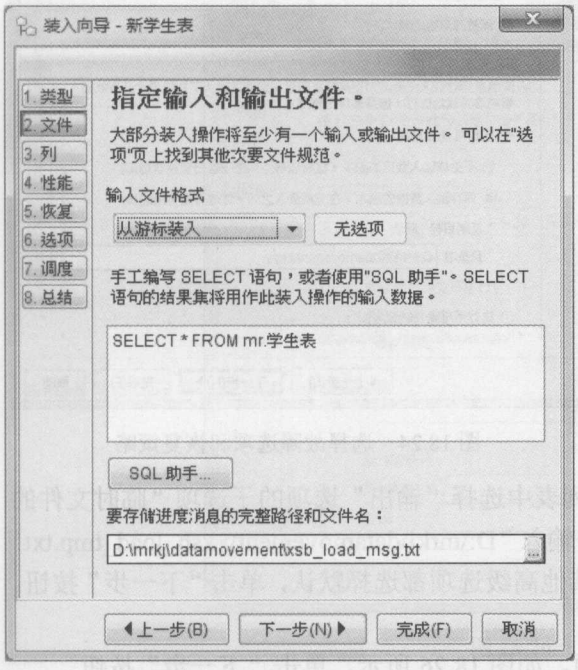


图 18.20　指定输入和输出文件

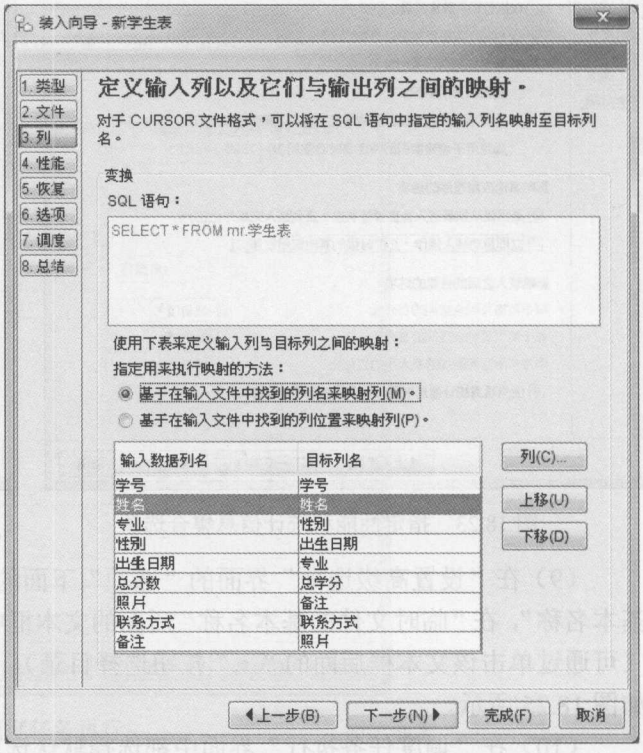


图 18.21　定义输入列以及它们与输出列之间的映射

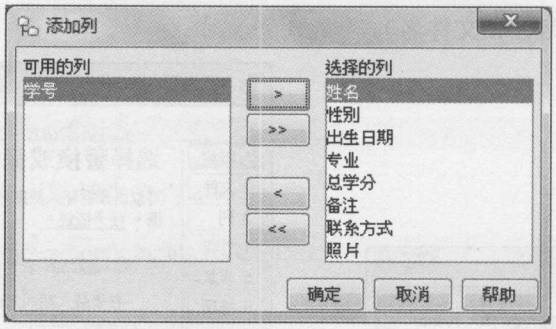


图 18.22　装入数据的目标列

（7）指定性能和统计信息集合选项，如图 18.23 所示。选中“使用现有统计信息概要文件来更新表统计信息”复选框，其他选择默认，单击“下一步”按钮。

（8）在“选择故障选项和恢复策略”界面的“正向恢复”栏中选中“执行可恢复的装入”单选按钮和该单选按钮下面的子单选按钮“保存输入数据的副本，在完成装入之后，该表将是可用的”，在“复制目标”后面的下拉列表框中选择“目录”，在“目录名”后面的文本框中输入“D:\mrkj\ datamovement\copy”（此目录必须已经存在），单击“下一步”按钮，如图 18.24 所示。

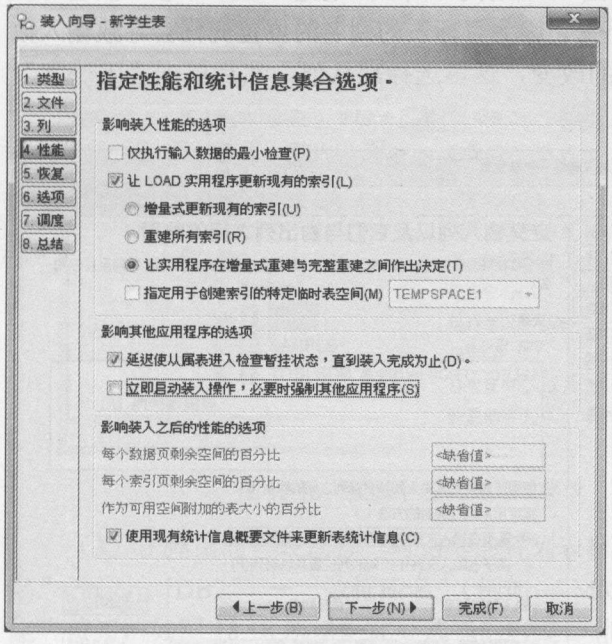


图 18.23　指定性能和统计信息集合选项

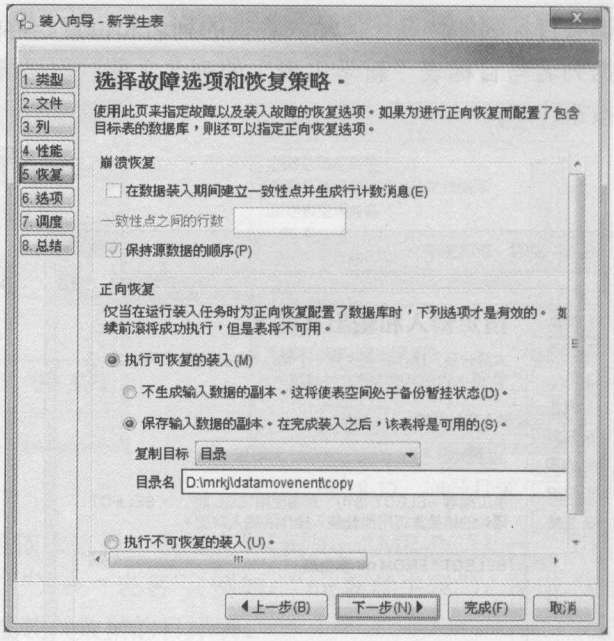


图 18.24　选择故障选项和恢复策略

（9）在“设置高级选项”界面的“选项”下面的列表中选择“输出”选项的子选项“临时文件的基本名称”，在“临时文件的基本名称”后面的文本框中输入“D:\mrkj\datamovement\xxsb_load_tmp.txt”（可通过单击该文本框后面的“...”按钮选择目录），其他高级选项都选择默认，单击“下一步”按钮，如图 18.25 所示。

（10）在“调度任务执行”界面中都选择默认选项，如图 18.26 所示，单击“下一步”按钮。

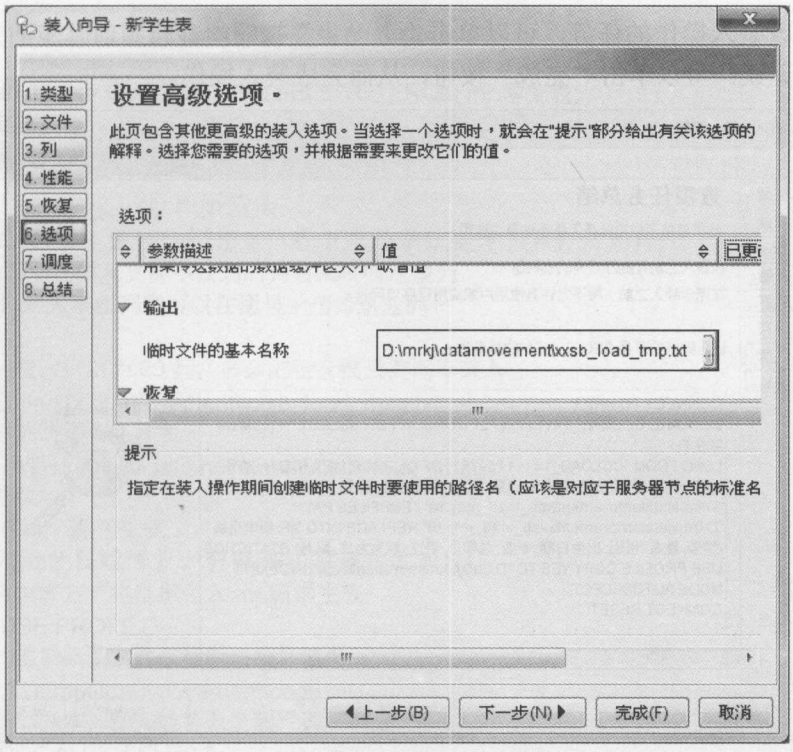


图 18.25　设置装入的高级选项

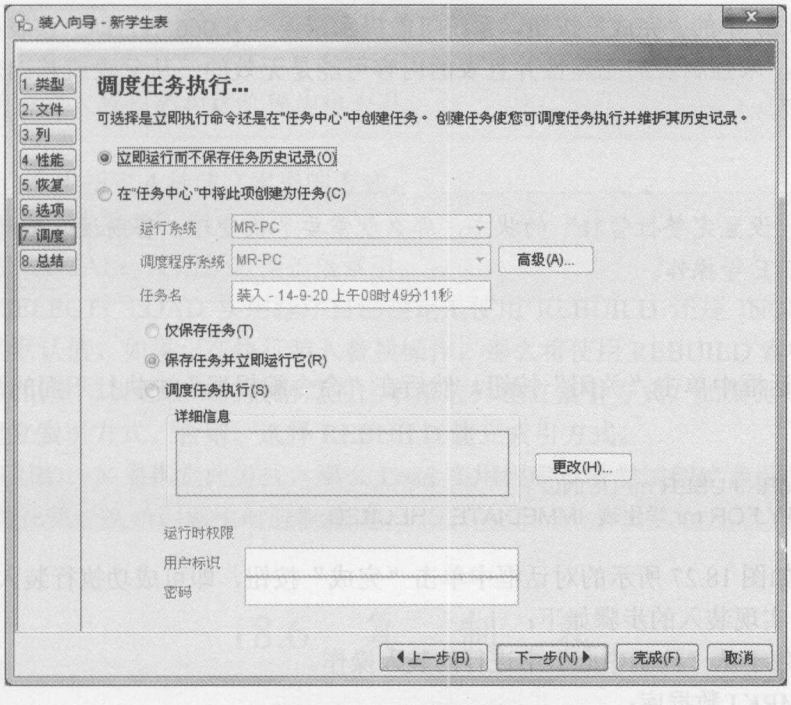


图 18.26　调度任务执行

（11）在如图 18.27 所示“查看任务总结”界面中，系统自动列出了前面装入操作对应的 SQL 命

令集合。如果要修改装入操作的任务，可以单击“上一步”按钮，并按照前几步的方法修改装入操作的任务。如果确认无误，可以单击“完成”按钮，从而完成装入操作。

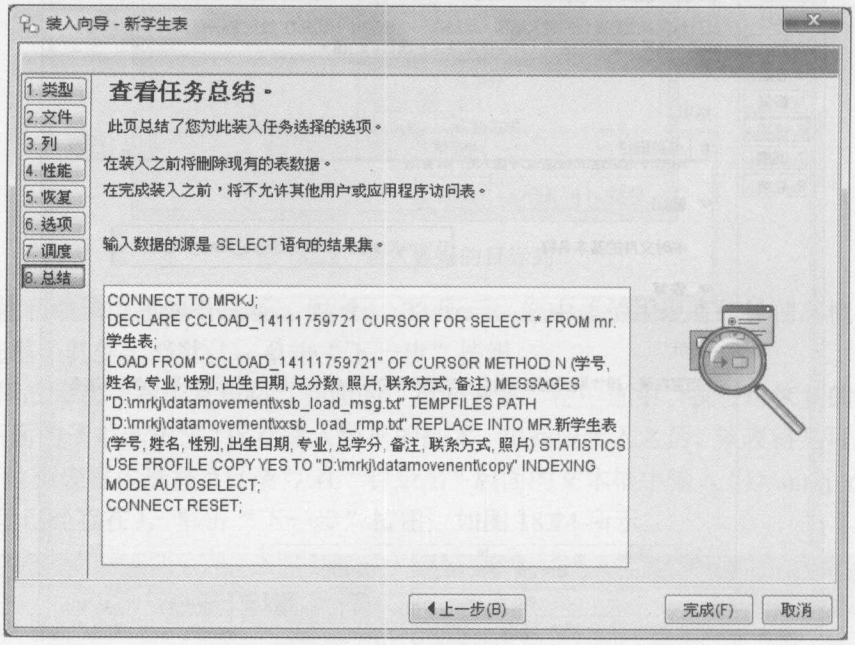


图 18.27　查看装入任务的总结

如果单击图 18.27 中的“完成”按钮，系统可能提示错误 SQL0668N，即“该表处于‘设置完整性暂挂无访问’状态。未强制表的完整性并且表的内容可能是无效的。从父表或基础表执行的操作也可能会接收到此错误”。

说明

如果表处于“设置完整性暂挂”的状态，那么就需要尽快处理，否则该表不能进行 SELECT、UPDATE、DELETE 等操作。

解决方法如下：

在错误提示对话框中单击“关闭”按钮，然后在“命令编辑器”中执行下面的命令来解除表的检查挂起状态：

```
CONNECT TO MRKJ USER mr USING 1;
SET INTEGERITY FOR mr.学生表 IMMEDIATE CHECKED;
```

之后，重新在如图 18.27 所示的对话框中单击“完成”按钮，即可成功执行装入命令。

用 LOAD 命令实现装入的步骤如下：

下面的命令等效于从“控制中心”中进行的装入操作。

（1）连接到 MRKJ 数据库：

```
CONNECT TO mrkj USER mr USING 1;
```

（2）定义一个光标，光标指向即将装入的数据：

```
DECLARE load_cursor CURSOR FOR SELECT * FROM mr.学生表;
```

（3）从光标类型文件中进行装入：

```
LOAD FROM load_cursor OF CURSOR
      METHOD N (学号,姓名,性别,出生日期,专业,总学分,备注,联系方式,照片)
      --N  表示基于在输入文件中找到的列名来映射这些列
      --如果指向装入某些列则可以在括号中选择这些列
ROWCOUNT 20
--设置要处理的最大行数为 20 行，多余的行会被丢弃而不装入
MESSAGES D:\mrkj\datamovement\xxsb_load_msg.txt
--存储进度消息的路径和文件名
TEMPFILES PATH D:\mrkj\datamovement\xxsb_load_tmp.txt
--临时文件的路径
REPLACE INTO mr.新学生表
(学号,姓名,性别,出生日期,专业,总分数,备注,联系方式,照片)
--采用替换表数据的方式将数据装入 mr.新学生表
STATISTICS USE PROFILE
--使用现有统计信息概要文件来更新表统计信息
COPY YES TO D:\mrkj\datamovement\copy
--保存输入数据的副本，在完成装入之后，该表将是可用的
INDEXING MODE AUTOSELECT;
--装入操作的构建阶段采用 AUTOSELECT 模式构建索引
```

说明

上述命令将在装入操作的构建阶段构建索引。

可在 LOAD 命令中指定 4 种建立索引的方式。

（1）REBUILD：将重建所有索引。

（2）INCREMENTAL：使用新数据扩展索引。

（3）AUTOSELECT：LOAD 实用程序自动决定是使用 REBUILD 还是 INCREMENTAL 方式，AUTOSELECT 是默认值。如果正在执行装入替换操作，那么将使用 REBUILD 索引建立方式。否则，根据表中的现有数据量与新近装入的数据量的比率来选择建立索引方式。如果比率非常大，那么选择 INCREMETAL 建立索引方式。否则，选择 REBUILD 建立索引方式。

（4）DEFERRED：如果指定此方式，那么 Load 实用程序不会尝试创建索引。索引将标记为需要刷新，并且可能会在第一次访问索引时强制重建。

18.6　复　制　表

【例 18.9】　在“控制中心”创建一个“课程表”的副本“课程表副本”。

注意

“课程表副本”表在复制操作之前不存在。

实现步骤如下：

（1）在“控制中心”对象视图中选择 MRKJ 数据库的“表”文件夹，在内容窗格中找到“课程表”，如图 18.28 所示，在该表单击鼠标右键，在弹出的快捷菜单中选择“复制”命令，从而打开“复制表-课程表”对话框。

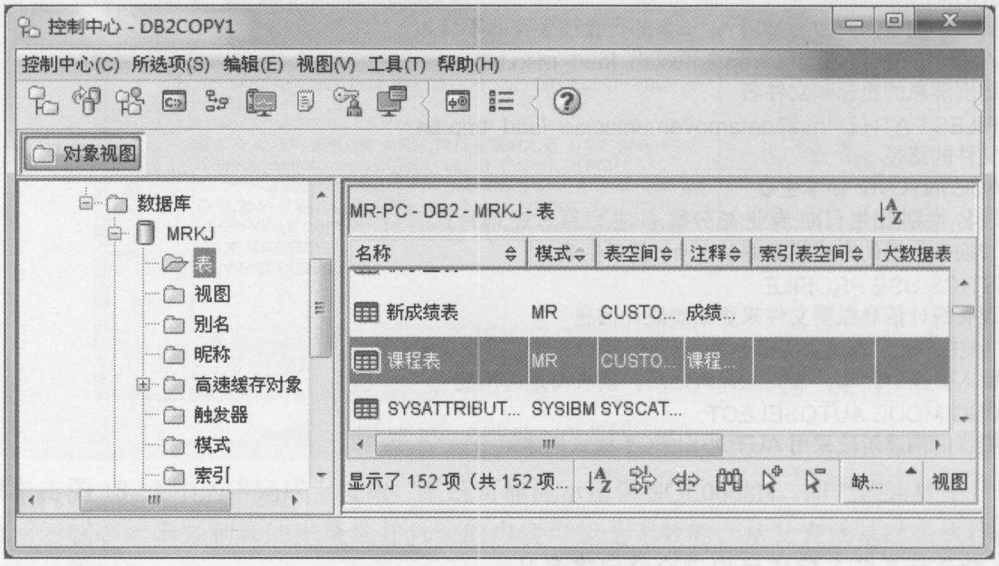


图 18.28　在内容窗格中找到“课程表”

（2）在如图 18.29 所示的“复制表-课程表”对话框中，在“系统”后面的列表框中选择目标表的系统名称（这里是 MR-PC），在“实例”后面的列表框中选择目标表的实例名称（这里是 DB2），在“数据库”后面的列表框中选择目标表的数据库名称（这里是 MRKJ），在“表模式”后面的列表框中选择目标表的表模式名称（这里是 MR），在“表名”后面的文本框中输入表副本的名称（这里是“课程表副本”，此表当前不存在），在“表空间”后面的列表框中选择目标表的表空间名称（这里是 CUSTOMSPACE1），输入完成后，可以单击“显示命令”按钮查看生成的 SQL 命令，如图 18.30 所示。

从图 18.30 可以看出，复制表的过程实质上就是先用 EXPORT 命令导出源表数据，然后用 IMPORT 命令将导出的数据导入到副本表中。单击“关闭”按钮，再单击“确定”按钮即可完成复制。

注意

复制表向导不支持 XML 列，所以如果源表含有 XML 列则不能成功复制表。例如，不能用向导复制“学生表”，因为学生表中的联系方式这一列是含有 XML 列的。

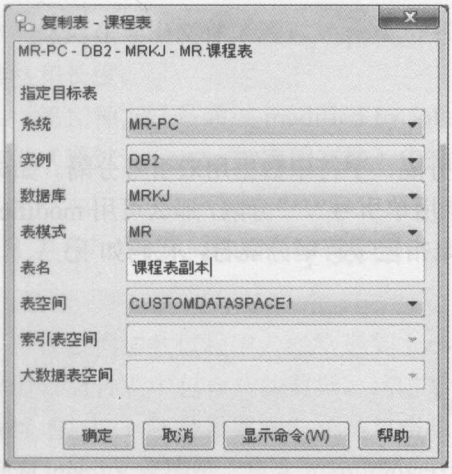


图 18.29　“复制表-课程表”对话框

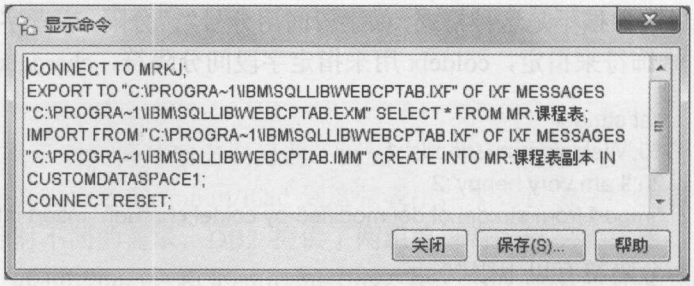


图 18.30　复制表的 SQL 命令

18.7　常 见 问 题

下面给读者列举几个常见的问题。

18.7.1　加载的数据是 Excel 格式

需要处理 Excel 数据，可将其另存为“CSV（逗号分隔）”格式，然后按照前述方法导入，如图 18.31 所示。

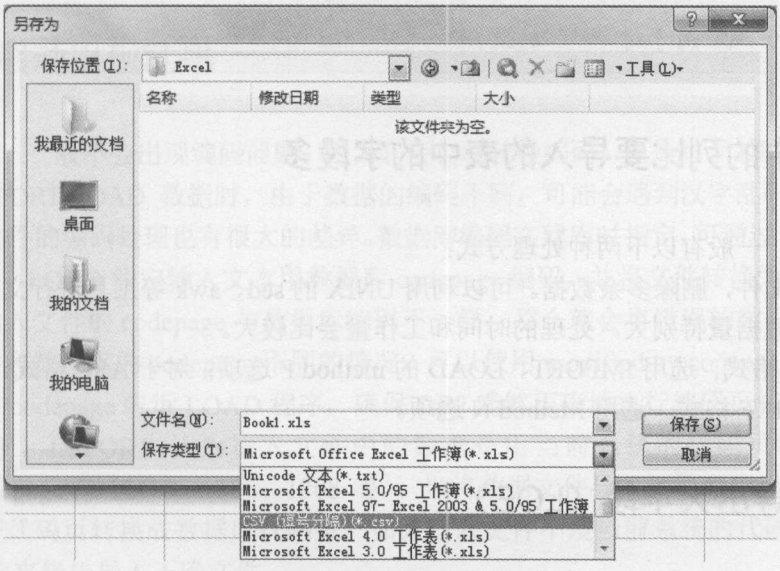


图 18.31　另存为“CSV（逗号分隔）”格式

18.7.2 导出/加载的数据不是逗号/双引号分隔

在默认情况下，对于 DEL 格式，每行数据字段之间用逗号分隔，字符串数据用双引号分隔。如果提供的数据不是这种格式，如字段间用分号“;”分隔，或者字符串用单引号“‘’”分隔，那么可用 modified by 修饰符来指定，coldelx 用来指定字段间分隔符，chardelx 用来指定字符串分隔符，代码如下：

```
cat stu.del
10;'Welcome my friends';1
20;'I am very heppy';2
"import from stu.del of del modified by coldel;chardel' ' insert into stu"
```

分隔符有以下限制：

- ☑ 用户要确保数据中不能包含分隔符，如果存在，会出现各种异常。
- ☑ 分隔符不能是换行符、回车符、0x00 或空格。
- ☑ 在 DBCS 环境下（如中文系统），不能用“|”符号作为分隔符。
- ☑ 如果 DEL 文件通过某个特殊字符分隔，而 IMPORT、LOAD、EXPORT 命令中通过键盘无法敲入时，可转换为十六进制表示，如 0x7c 代表“|”。

对于 decimal 值，默认的导出格式是在前面有+号，如果位数不够定义的长度，则用 0 补齐。如果用户不希望保留+号，或者去掉前面的 0，DB2 提供了两个修饰符选项，一个是 delplusblank（del plus blank，即+换成空格），另一个是 striplzeros（strip zero，去掉空格）。

DB2 对于日期格式的支持也比较灵活，如果用户指定了导出的时间格式，那么可以通过 modified by timestampformat 来指定。

如果用户要求导出的时间按照“年/月/日 时:分:秒:毫秒”的格式，可在 EXPORT 命令中指定。

注意

YYYY 前面的“\”用于对“‘”转义。

18.7.3 文件中的列比要导入的表中的字段多

针对这一情况，一般有以下两种处理方式：

- ☑ 修改数据文件，删除多余数据。可以利用 UNIX 的 sed、awk 等工具进行文件内容处理，但是如果文件数据量特别大，处理的时间和工作量会比较大。
- ☑ 对于 DEL 格式，选用 IMPORT、LOAD 的 method P 选项；对于 ASC 格式，选用 Method L 选项；对于 IXF 格式，选用 Method N 选项。

18.7.4 导入/导出大字段（LOB）

第一种方法是通过 modified by lobsinfile 将每个大对象字段的所有行数据存放在 lobs to 指定的文件

中，然后在导出文件（通过 export to 指定的文件）中通过指针来定位大对象数据在指定文件中的配置偏移和长度。

第二种方法是通过 modified by lobsinsepfiles 将每一行的大对象数据存放在独立的文件中，在导出文件（通过 export to 指定的文件）中指定文件名。

18.7.5　导入 identity 数据

在数据加载过程中，经常遇到 identity 表示列数据处理出现问题。例如，某张表含有 identity 自增字段，而且表中包含一些数据。这时需要把某文件数据 import/load 到这张表中，那么自增字段的值是继续增加呢？还是重置呢？还是忽略呢？针对不同的需求，DB2 提供了两组修饰符来解决此问题。一组为 identity 修饰符，包含 identityoverride、identityignore 和 identitymissing，用于当表字段有 generated always as identity 或 generated by default as identity 自增列时数据加载的处理方法；另外一组为 generated 修饰符，包含 generatedoverride、generatedignore 和 generatedmissing ，用于当表字段有 generated always as<expression>时数据加载的处理方法。

Identityoverride 表示使用数据文件中提供的值，特别适用于系统间数据迁移时，需要保留 identity 列值的情况；identitymissing 表示当数据文件中不包含自增值时，import/load 数据产生自增值；identityignore 命令表示当数据文件中包含自增值时，import/load 命令忽略该值，并生成自增值。所以可以根据数据文件和用户需求，决定采用哪个修饰符。

注意

Identityoverride 修饰符只适用于 generated always，不支持 generated by default。Identityoverride 不能在 IMPORT 命令中使用，只在 LOAD 中支持。

18.7.6　数据移动出现乱码

对于 IXF 格式，一般不会出现编码问题。因为它会自动转换编码。但是对于 DEL 格式，当在不同系统间 EXPORT/IMPORT/LOAD 数据时，由于数据的编码不同，可能会遇到汉字乱码问题，而 IMPORT 和 LOAD 对数据文件的编码处理也有很大的差异。数据库编码在建库时指定，可通过 db2 get db cfg 查看。

在默认情况下，LOAD 假定输入文本用数据库 codepage 编码，并将文件转换为数据库 codepage 编码来导入。如果输入文件的 codepage 与数据库编码不一样，那么就会将错误的数据装载到数据库中。因此对于数据文件和数据库的 codepage 不同的情况，可以使用 modified by codepage=<codepage>选项，显式地将数据文件 codepage 告诉 LOAD 程序，确保 DB2 能够正确地执行编码页转换。

在默认情况下，IMPORT 认为输入文件中的数据是用当前系统的代码页编码（通过 db2set db2codepage=<codepage>设置，如 1386、1208 等）。当将数据文件导入到数据库时，DB2 会自动将数据文件从当前系统代码页转换成数据库代码页。如果输入文件不是当前系统的代码页编码，也可以通过 codepage 修饰符来轻松导入正确文件。

18.7.7 表数据从一个表空间迁移到另一个表空间

表数据在不同表空间的迁移是很多客户遇到的问题，原因有以下几个：

☑ 表空间页大小初始设置是 4KB 页，随着业务需求的变化，某张表（假设为 t1）的一行记录大小需要超过 4KB，当前表空间页大小无法满足需求。

☑ 在 V8 版本，4KB 页大小的表空间最大限制为 64GB。当数据量达到此限制时是无法插入数据的。

☑ 出于性能考虑，需要将某些表从一个表空间迁移到另一个表空间。

对于此类问题，典型做法是先创建一个表空间，在此表空间创建与原表相同的表结构（假设为 t2），然后将原表 t1 数据导出，再导入到 t2 表，完成测试后，将 t1 表改名或删除，将 t2 表改名为 t1。如果要迁移的表很多，可考虑写成脚本。注意：如果在 t1 表上有主外键约束等，则需要先将其删除，否则无法 rename。

例如：

```
create tablespace ts2;
create table t2 like t1 in ts2;
export to t1.del of del select * from t1;
load from t1.del of del insert into t2;
rename t1 to t1_bak;
rename t2 to t1;
```

可以看到，上述解决方案操作比较繁琐，而且要求在操作时 t1 是离线的，即不允许应用进行增、删、改操作，因此只能在特定的运维时间窗口内操作，如果表数据量很大，在指定的时间无法完成的话，则对应用的影响较大。

为了增加系统的可用性，V9.7 版本引入了在线数据迁移方案，通过 ADMIN_MOVE_TABLE 存储过程将表数据从一个表空间迁移到另外一个表空间，迁移过程中保持对原表数据的持续访问。

ADMIN_MOVE_TABLE 过程有两种等价的调用方式。第一种是提供目标表的数据、索引、大对象表空间名；第二种是先建一个表，建表时指定数据、索引、大对象表空间，在调用时提供表名而不需要提供表空间名，迁移完成后，表名替换为原表名。

对于在线表迁移操作，需要以下几个表。

☑ 原表（source table）：要迁移的表名。

☑ 目标表（target table）：迁移的目标，如果是第一种调用，则这张表会在指定的表空间自动创建，如果是第二种调用，目标表已经建好，作为参数传给存储过程即可。在迁移过程中，仍然可以在原表上做数据的 INSERT、UPDATE 和 DELETE 操作，通过 Tigger 机制来捕获变化数据，并同步到目标表上，为此需要创建第三张表。

☑ 中间表（staing table）：记录迁移过程中在原表上产生的更新数据，这些变化数据随后被同步到目标表上。迁移完成后，这张表会删除。

在线迁移操作包括 5 个阶段：INIT、COPY、REPLAY、SWAP 和 CLEAN UP，下面分别进行介绍。

☑ INIT 阶段：它又叫作初始化阶段，将验证调用者是否有权限进行表迁移；根据调用方法验证或创建目标表；创建中间表。在原表上创建 4 个触发器，这样对原表的 INSERT、UPDATE、

DELETE 操作都将触发触发器，并将变化记录到中间表中，对 update 操作，更新前和更新后的值都将记录。

- ☑ COPY 阶段：复制阶段就是从原表复制数据到目标表，当复制时原表仍然可访问。复制数据的方式有两种，一种是 insert<Target>select * from <Source>方式；另一种是采用 load…from cursor 的方式。复制的最后，会在目标表上创建和原表一致的索引。复制过程中对原表的变更都将记录到中间表中。
- ☑ REPLAY 阶段：REPLAY 阶段会遍历中间表，将数据写到目标表上。根据原数据的变更情况，REPLAY 可能需要做很多次。
- ☑ SWAP 阶段：此阶段将交换原表和目标表表名，然后删除原表。在交换表名之前，需要确保两边数据完全一致。因此，在最后一次 REPLAY 时，要在原表和目标表上加锁，确保没有新数据产生，这样保证了两表数据完全一致。然后删除原表，重命名目标表到原表，并释放锁。
- ☑ CLEAN UP 阶段：它又叫最后的清理阶段，它将清除一些临时对象，删除第一步创建的触发器并删除中间表。

表迁移限制，有些情况下不支持在线表迁移过程，可以根据 reason code 查找错误原因，以下是一些常见的限制：

- ☑ 原表上不能有主外键约束，主表或依赖表都不可以有主外键约束，可以通过 db2look 或系统表查找约束，并在迁移前删除这些约束，待迁移后再重新建立这些约束。
- ☑ 如果表上有事件监控器则不能迁移。
- ☑ 如果表上有 LOB、XML、LONG 类型字段，则要求表上必须有唯一性索引。
- ☑ 处于 set integrity pending 状态的表不能迁移。

18.8 小 结

本章主要对 DB2 中的数据移动进行详细讲解，具体讲解过程中，首先介绍了数据移动的概念，让读者对什么是数据移动有一个清晰的了解；然后讲解了文件格式、如何导出数据、如何导入数据以及装入数据等内容；最后还讲解了如何复制表等内容。通过本章的学习，读者应该熟练掌握数据移动的使用，并能够使用数据移动解决数据库开发中遇到的问题。

18.9 实践与练习

（1）数据移动的作用是什么？（**答案位置：光盘\TM\sl\18\7**）

（2）有哪几种文件格式，分别是什么？（**答案位置：光盘\TM\sl\18\8**）

（3）说出装入过程分为几个阶段以及各阶段的含义？（**答案位置：光盘\TM\sl\18\9**）

第19章

事务与锁

（视频讲解：27分钟）

现在的IT系统对多用户并发性能提出了很高的要求，那么如何保证多用户并发情况下数据的一致性呢？答案就是事务、日志、锁和隔离级别。DB2 UDB系统主要是通过“锁”和“隔离级别”这两个工具来控制并发连接，维护数据库中的数据在高并发的环境下的安全。

通过阅读本章，您可以：

- 了解数据库事务
- 了解事务日志记录
- 掌握并发性控制
- 掌握锁
- 了解隔离级别

19.1 数据库事务

数据库事务（Database Transaction）是由一系列语句构成的逻辑单元，要么完全执行，要么完全不执行。例如，网上购物的一次交易，其付款过程至少包括以下几步数据库操作：

（1）更新客户所购商品的库存信息。

（2）保存客户付款信息——可能包括与银行系统的交互。

（3）生成订单并且保存到数据库中。

（4）更新用户相关信息，例如购物数量等。

通常情况下，这些操作将顺利进行，最终交易成功，与交易相关的所有数据库信息也成功地更新。但是，如果在这一系列过程中任何一个环节出了差错，例如在更新商品库存信息时发生异常、该顾客银行卡账户存款不足等，都将导致交易失败。一旦交易失败，数据库中所有信息都必须保持交易前的状态不变，例如最后一步更新用户信息时失败而导致交易失败，那么必须保证这笔失败的交易不影响数据库的状态，库存信息没有被更新、用户也没有付款，订单也没有生成。否则，数据库的信息将会一片混乱而不可预测。

数据库事务正是保证这种情况下交易的平稳性和可预测性的技术，它是作为单个逻辑工作单元执行的一系列操作。

一个逻辑工作单元必须有 4 个属性，称为原子性、一致性、隔离性和持久性（ACID），只有这样才能成为一个事务，下面分别对这 4 个属性进行介绍。

（1）原子性：事务必须是原子工作单元；对于其数据修改，要么全都执行，要么全都不执行。

（2）一致性：事务在完成时，必须使所有的数据都保持一致状态。在相关数据库中，所有规则都必须应用于事务的修改，以保持所有数据的完整性。事务结束时，所有的内部数据结构（如 B 树索引或双向链表）都必须是正确的。

（3）隔离性：由并发事务所做的修改必须与任何其他并发事务所做的修改隔离。事务识别数据时数据所处的状态，要么是另一并发事务修改它之前的状态，要么是第二个事务修改它之后的状态，事务不会识别中间状态的数据，这称为可串行性，因为它能够重新装载起始数据，并且重播一系列事务，以使数据结束时的状态与原始事务执行的状态相同。

（4）持久性：事务完成之后，它对于系统的影响是永久性的。该修改即使出现系统保障也将一直保持。

19.2 事务日志记录

DB2 确保数据一致性和可恢复性所使用的主要机制之一是事务日志记录。

DB2 有两种可用的事务日志文件——主日志文件和辅助日志文件，其详细介绍请参见第 15 章。

DB2 有两种可用的事务日志记录模式——循环（Circular）日志记录和归档（Archive）日志记录，

其详细介绍请参见第 15 章。

19.2.1 控制事务日志记录的参数

通过数据库配置参数可在数据库级别上控制事务日志记录。

以下是影响事务日志记录的参数。

- ☑ LOGRETAIN：从 DB2 V9.5 开始，就不推荐使用此参数。此参数确定是否保留活动日志文件以及这些文件是否可用于前滚恢复。DB2 将忽略对此配置参数指定的任何值。
- ☑ USEREXIT：从 DB2 V9.5 起，就不推荐使用此参数。DB2 将忽略对此配置参数指定的任何值。如果启用此参数，那么不管 LOGRETAIN 参数的设置如何，都将执行日志，保留日志记录操作。此参数也指示应用使用用户出口程序来归档和检索日志文件。
- ☑ LOGPRIMARY：此参数指定要预分配的主日志文件数。主日志文件建立分配给恢复日志文件的固定存储器数量，取值为 2~256，默认值为 3。
- ☑ LOGSECOND：此参数指定（仅需要时）创建并用于恢复日志文件的辅助日志文件的数目。取值为-1 或 0~254，默认值为 2。如果将 LOGSECOND 设置为-1，那么数据库将配置为具有无限活动日志空间。如果将 LOGSECOND 设置为-1，那么仍使用 LOGPRIMARY 和 LOGFILSIZ 配置参数来指定 DB2 应在活动日志路径中保留多少个日志文件。如果日志路径是原始设备，那么必须配置 OVERFLOVLOGPATH 配置参数，以便将 LOGSECOND 设置为-1。
- ☑ OVERFLOWLOGPATH：溢出日志路径配置参数。此参数指定一个位置，DB2 将在该位置中查找前滚操作所需的日志文件，并使用该位置存储从归档中检索到的活动日志文件。它还提供了查找和存储使用 DB2READLOG API 所需的日志文件的位置。取值可以为任何有效路径，默认值为 NULL。要设置 OVERFLOWLOGPATH，指定一个最长 242 个字节的字符串。该字符串必须指向路径名，并且它必须为标准路径名，而不是相对路径名。该路径名必须是目录而不是原始设备。
- ☑ LOGFILSZ：该参数确定了每个已配置日志的页数量。一页的大小是 4KB。
- ☑ LOGBUFSZ：该参数指定数据库共享内存的数量，在将日志记录写到磁盘之前，用该共享内存做完这些记录的缓冲区。
- ☑ MINCOMMIT：该参数指定允许延迟将日志记录写到磁盘，直到已经执行了所规定的最小数量的提交。
- ☑ NEWLOGPATH：数据库日志最初创建在名为 SQLOGDIR 的目录中，该目录是数据库目录的子目录。可以通过将该配置参数的值更改成指向另一个目录或设备来更改放置活动日志和将来归档日志的位置。如果将数据库配置成进行前滚恢复，那么就不会将当前存储在数据库日志路径目录中的归档日志移到新的位置中。
- ☑ LOGARCHMETH1：此参数设置主日志归档方法和目标。
- ☑ LOGARCHMETH2：此参数设置辅助日志归档方法和目标。如果指定了此路径，那么日志文件将被同时归档至此目标和由 LOGARCHMETH1 数据库配置参数指定的目标。

注意

日志文件的数量（LOGPRIMARY+LOGSECOND）乘以以字节为单位的每个日志文件的大小（LOGFILSIZ*4096）必须小于 512GB（DB2 V9.5 以上为 512GB，早前版本中最大为 256GB）。

【例 19.1】　修改日志参数。

可通过控制中心和 SQL 命令两种方式修改日志参数。

通过控制中心修改日志参数的实现步骤如下：

（1）在“控制中心”对象视图中的数据库文件夹中选择要修改日志参数的数据库（例如 MRKJ），在该数据库的右键快捷菜单中选择“配置参数”命令，从而打开“数据库配置”对话框。在该对话框中的“日志”目录下列出了可以修改的各种日志参数，如图 19.1 所示。



图 19.1　配置数据库的日志参数

这时可以看到在第 18 章用 SQL 语句修改的 LOGARCHMETH1 参数，此语句将循环日志模式修改为归档日志模式，且将归档日志位置指定为“d:\mrkj\archive”，该 SQL 语句为：

```
UPDATE DB CFG FOR mrkj USING LOGARCHEMETH1 "DISK:d:\mrkj\archive";
```

（2）选择要修改的日志参数所在的行，并单击某个参数后面的文本框后面的“…”按钮，单击此按钮后会弹出“更改数据库配置参数”对话框。例如，选择了 LOGARCHMETH1 日志参数，并单击 LOGARCHMETH1 后面的文本框后面的“…”按钮，单击此按钮后会弹出“更改数据库配置参数”对话框，如图 19.2 所示。

（3）在如图 19.2 所示的“更改数据库配置参数”对话框中可以选中某个日志归档单选按钮（例如，选择 OFF 表示关闭 LOGARCHMETH1 的值所指的目录中的归档日志，如果 LOGARCHMETH2 日志参数的值也为 OFF，则关闭归档日志模式并返回循环日志模式），然后单击“确定”按钮确认修改。



图 19.2　修改 LOGARCHMETH1 日志参数

（4）修改后的日志参数值出现在如图 19.3 所示的对话框中的“暂挂值”列中，单击“确定”按钮，完成日志参数的修改。修改的参数在可用时或重启数据库时生效。

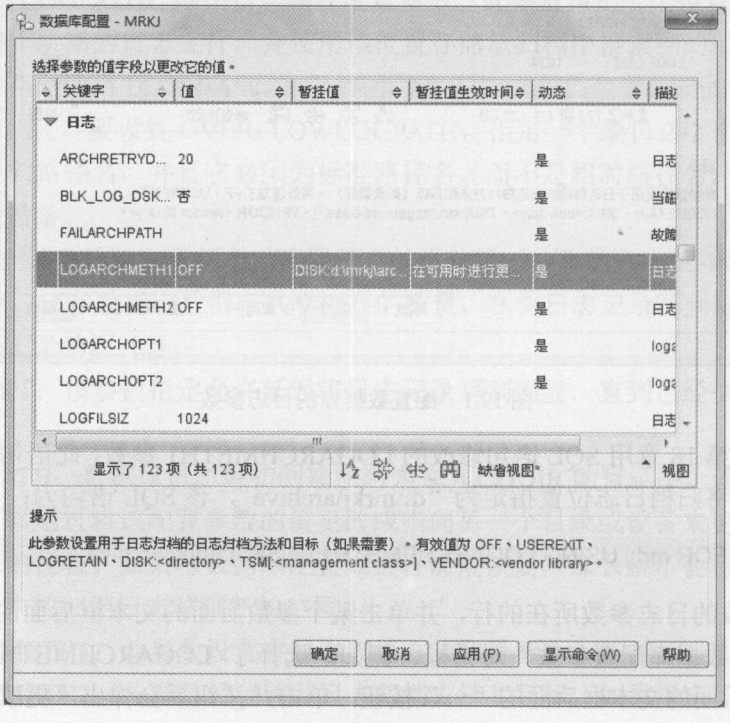


图 19.3　配置数据库的日志参数

修改日志参数的命令格式为：

```
UPDATE  <DATABASE | DB> <CONFIGURATION | CONFIG | CFG> [FOR database-alias]
[DBPARTITIONNUM db-partition-NUM] USING config-keyword
<value | value AUTOMATIC | AUTOMATIC | MANUAL>
[IMMEDIATE | DEFERRED]
```

参数说明如下。

- ☑ DBPARTITIONNUM db-partition-num：如果是分布式数据库，该子句指定分区号。
- ☑ Config-keyword：要配置的数据库参数名，该参数可以是日志参数或数据库的其他参数。
- ☑ AUTOMATIC：允许 DB2 根据当前资源的需要自动调整这些参数值。
- ☑ MANUAL：手动调整参数，不允许系统自动调整参数值。
- ☑ IMMEDIATE | DEFERRED：默认值是 IMMEDIATE，表示立刻生效。DEFERRED 表示仅在配置文件中修改参数，以便所做的修改在下次激活数据库时生效。

19.2.2　日志文件存储的位置

默认情况下，日志文件存储在以下目录中：C:\【实例名】\NODE0000\SQL00001\SQLOGDIR。

在上述目录路径中，对于所创建的每个数据库，会有一个 SQL×××××（“××××××”是以 0 开头的数字）目录。如果在 DB2 实例中有多个物理数据库，就很难知道哪个 SQL×××××目录属于哪个数据库。要解决这个问题，需要以下命令：

```
list database directory on <drive_name>;
```

在归档日志记录模式下，日志文件的存储位置为 LOGARCHMETH1 和 LOGARCHMETH2 日志参数所指的位置。

【例 19.2】　查询 D 盘上驻留的所有数据库（**实例位置：光盘\TM\sl\19\1**）。

具体语句如下：

```
list database directory on d: ;
```

说明

如果 MRKJ 数据库在 D 盘，则上述命令返回结果中可以看到 MRKJ 数据库的别名、名称、目录、注释、数据库分区等信息，如图 19.4 所示。可使用类似命令查看其他磁盘上驻留的 DB2 数据库。

```
数据库别名                  = MRKJ
数据库名称                    = MRKJ
数据库目录                  = SQL00003
数据库发行版级别              = d.00
注释                      = 学生成绩数据库
目录条目类型                  = 本地
目录数据库分区号              = 0
数据库分区号                  = 0
```

图 19.4　查询数据库信息

19.3 并发性控制

DBMS 的主要优点之一就是可以在多个用户和多个应用程序中共享数据，所以数据库系统应该提供一种管理并发访问数据库的方法。

如果没有控制并发性的有效方法，就可能损害数据的完整性和一致性。DBMS 必须保护数据库，防止发生下列状况。

（1）丢失更新。假设应用程序 A 和应用程序 B 同时读取数据库中的同一行，并且都为其中某一列计算新值。如果应用程序 A 先用其新值更新该行，随后应用程序 B 又更新同一行，那么第一次的更新（由应用程序 A 执行的）就会丢失。

（2）脏读。如果一个事务在提交操作结果之前，另一个事务可以看到该结果，就会发生如图 19.5 所示的情况。

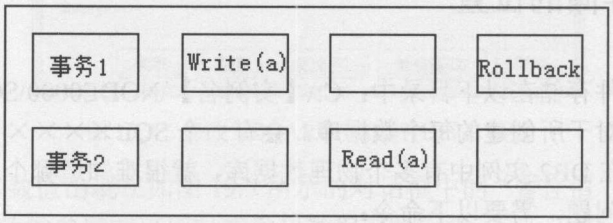


图 19.5 发生脏读的示例

图 19.5 中，如果事务 2 读取了事务 1 所写的值，则“脏读”就发生了。

（3）不可重复读（也称为模糊读（Fuzzy Read））。如果一个事务提交结果之前，另一个事务可以修改和删除它，就会发生如图 19.6 所示的情况。

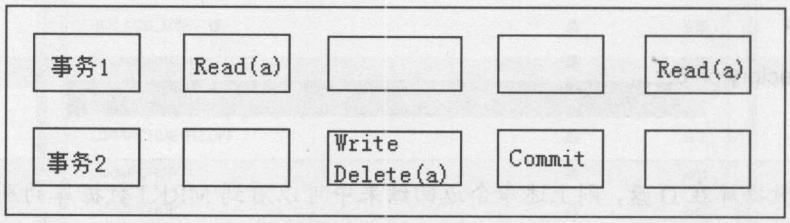


图 19.6 不可重复读

图 19.6 中，如果事务 1 每次从 Read 上获取的是不同的结果，就会发生“不可重复读”。

（4）幻像读。如果一个事务在提交查询结果之前，另一个事务可以更改该结果，就会发生这种情况，如图 19.7 所示。

图 19.7 中，如果事务 1 每次从 Select 上获取的是不同的结果，就会发生“幻像读”。

在维护数据完整性的同时，提供多个应用程序同时访问数据的能力称为并发性控制。

DB2 UDB 系统主要通过“锁”和“隔离级别”这两个主要的工具来控制并发连接，维护数据库中的数据在高并发的环境下的安全。

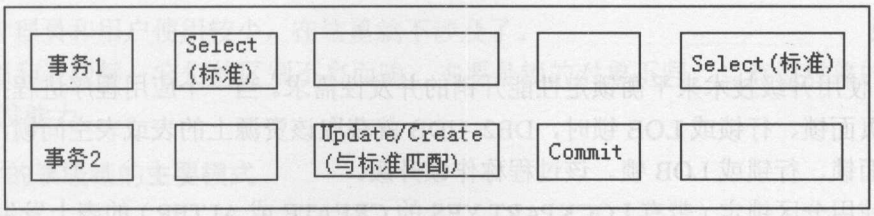


图 19.7　幻像读

19.4　锁

DB2 需要一种机制来解决多个事务并发处理时数据的一致性和完整性。这种机制就是锁。DB2 管理器会根据这个锁的类型来决定其他事务是否可以获取该锁。当该事务完成后，就会释放锁资源，其他事务就可以使用这个资源。

19.4.1　锁的基本概念

1. 锁

数据库中的锁是指一种软件机制，用来指示某个用户（即进程会话，下同）已经占用了某种资源，从而防止其他用户做出影响该用户的数据修改或导致数据库数据的非完整性和非一致性的操作。这种资源主要是指用户可以操作的数据行、索引以及数据表等。

2. 锁模式

最常用的两种锁模式是共享锁和排他锁。共享锁与只读操作有关联，这意味着持有该锁的应用程序可以读取数据，而其他应用程序也可以读取该数据。排他锁与写操作有关联，这意味着持有该锁的应用程序有资格更新数据，但在锁所有者完成更新（将修改提交给数据库）并释放该锁之前，其他应用程序无法使用该数据。在不同的锁定粒度上，共享锁和排他锁都有几种子类型的锁模式来实现锁定和并发性控制。

3. 锁粒度（LOCKSIZE）

LOCKSIZE 是 CREATE/ALTER TABLESPACE 语句的选项，在应用程序进程访问表空间中的表时，控制 DB2 UDB 获取何种类型的锁（即它决定该锁的“大小”，这有时也称作锁粒度）。该选项可以是 LOB、TABLESPACE、TABLE、PAGE、ROW 和 ANY。

CREATE TABLESPACE 语句的 LOCKSIZE 参数默认为 ANY。LOCKSIZE ANY 允许 DB2 UDB 选择锁大小。DB2 UDB 通常将 LOCKSIZE PAGE 用于非 LOB 的表，而将 LOCKSIZE TABLESPACE 用于 LOB 表。建议创建表空间时使用该默认值，除非有理由进行其他选择。如果选择修改 LOCKSIZE，那么就要根据使用该表空间的应用程序的性能监控结果和并发性特点来做决定。

4. 锁升级

DB2 UDB 使用升级技术来平衡锁定性能开销的并发性需求。当一个应用程序进程持有单个表或表空间上的大量页面锁、行锁或 LOB 锁时，DB2 UDB 就获取该资源上的表或表空间锁，然后释放该资源上以前的页面锁、行锁或 LOB 锁。该过程称作锁升级。

如果一个使用分区锁定（带有 LOCKPART YES 的 CREATE 或 ALTER）的表上发生锁升级，那么，就只升级当前被锁定的分区，未锁定的分区仍旧未锁定。一旦表空间中发生了锁升级，那么就要用表空间锁来锁定随后访问的未锁定分区。

在执行应用程序时，DB2 UDB 首先使用页面锁或行锁，并且只要该进程访问的页面或行相对较少，还会继续这样做。当该应用程序访问许多页面或行时，DB2 UDB 将变为使用表锁、表空间锁或分区锁。调用锁升级的准确时间是由 LOCKSIZE 和 LOCKMAX 的值决定的。

5. 暂挂

若一个应用程序进程请求一个锁，而该锁已被另一应用程序进程锁拥有，且不能共享，此时，就称该进程为暂挂的。请求应用程序会被挂起，也就是它将暂时停止运行。锁请求的优先次序是：将新来的锁请求按照接收次序进行排队，已经持有锁的应用程序的请求以及进行锁提升的请求要比新应用程序的请求先得到服务。而在那些分组中，请求次序则为“先进先出（First In，First Out，FIFO）”。

6. 超时

当应用程序处于暂挂状态超过了预设的一段时间间隔，那么就要终止该应用程序。该应用程序被称为已经超时。在终止该应用程序之前，会在 SQLCA 中收到一条合适的错误消息。SQLCA（SQL 通信域）是 SQL 应用程序预留的一块大小固定的存储区域，用于从 DB2 UDB 向程序传递条件代码和其他信息。但是，COMMIT 和 ROLLBACK 这两种操作不能超时。

7. 死锁

当多个应用程序每个都持有另一应用程序所需资源上的锁，没有这些资源，那些应用程序都无法继续完成其工作时，这时就会出现死锁状况。

以下是一个简单的死锁场景：

（1）应用程序 A 访问表 T，并请求页面 X 上的排他（非可共享的）锁。

（2）应用程序 B 访问表 T，并请求页面 Y 上的排他锁。

（3）应用程序 A 请求页面 Y 上的锁，同时仍然持有页面 X 上的排他锁。应用程序 A 将被挂起，因为应用程序 B 具有页面 Y 上的排他锁。

（4）应用程序 B 请求页面 X 上的锁，同时仍然持有页面 Y 上的排他锁。应用程序 B 将被挂起，因为应用程序 A 具有页面 X 上的排他锁。

这是一种僵持情况。应用程序 A 和 B 都无法继续工作。

19.4.2 行级锁和表级锁的模式

DB2 UDB 中，锁的主要作用对象是表和行，其他如表空间和索引也是锁的对象，但是因为其大多

为系统控制，管理员和用户使用较少，在这里就不涉及了。

对于行级锁和表级锁，它们的区别不言而喻，主要是锁的对象不同。当然锁对象的不同连带也会影响 DB2 的并发能力。

1. DB2 中的表级锁的主要模式

（1）S 锁（共享，Share）：锁拥有者和任何其他并发的事务都可以读（但不能更改）锁定的表中的数据。如果是共享锁锁定的，则其他并发事务可以读取数据，但不能对它进行更改。

（2）X 锁（互斥，独占，Exclusive）：锁的拥有者可以读取和更改被锁定的表或行中的数据。如果获取了互斥锁，则只允许使用“未提交的读”隔离级别的应用程序访问被锁定的表。对于用 INSERT、UPDATE 和/或 DELETE 语句操作的数据资源，将获取互斥锁。

（3）U 锁（更新，Update）：锁的拥有者可以更新被锁定表中的数据，并且锁的拥有者在它所有更新的任何行上自动获得 X 锁。其他并发的应用程序可以读取但不能更新被锁定表中的数据。

（4）IN 锁（无意图锁，Intent None）：不需要行锁，该锁的拥有者可以读表中的任何数据，包括其他事务尚未提交的数据，但不能对表中的数据进行更改。

（5）IS 锁（意向共享，Intent Share）：锁的拥有者可以读取锁定的表中的数据，但不能更改这些数据。同样，因为锁的拥有者不获取行级别锁，所以，其他并发的应用程序仍可以读取和更改表中的数据（当事务拥有表上的意向共享锁时，就在它读取的每个行上进行共享锁定）。当事务不传达更新表中行的意图时，就获取这种锁。

（6）IX 锁（意向互斥，Intent Exclusive）：锁拥有者和任何其他并发的应用程序都可以读取和更改被锁定表中的数据。当锁拥有者从表读取数据时，它在所读取的每一行上获取一个 S 锁，而在它更新的每一行上获取 U 锁和 X 锁。其他并发的应用程序可以读取和更新锁定的表。当事务传达更新表中行的意图时，就获取这种锁（SELECT FOR UPDATE、UPDATE…WHERE 和 INSERT 语句传达更新的意图）。

（7）SIX 锁（带意向互斥的共享，Share With Intent Exclusive）：此锁是被锁转换的产物，表示应用连接拥有 S 和 IX 锁的特性。

（8）Z 锁（超级互斥，Super Exclusive）：也称超级独占锁，锁的拥有者可以更改表、删除表、创建索引或删除索引。拥有此锁的应用连接对该表有完全的控制权。当事务尝试执行上述任何一种操作时，表上就自动获得这种锁。在除去这个锁之前，不允许其他并发事务读取或更新该表。

在这里主要看一下 IS/IX/SIX 这 3 个锁。在这 3 个锁中，IS/IX 本身并不具备使得应用连接可以读取或更新记录的能力，应用连接要读取和更新记录时，需要再得到相应的行级锁；反之亦然，任何应用要获得行级锁操作数据记录之前，也必须获得某个相应的表级锁。SIX 锁也是类似的情况。这就是为什么在很多情况下使用的是行级锁，但是用快照（SNAPSHOT）等工具却能够看到表级锁存在的原因。

2. 行级锁的类型

行级锁的类型如表 19.1 所示。

表 19.1 行级锁的类型

行级锁简称	行级锁全程	预先获得的表级锁的最低要求
S 锁	Share	IS 锁
U 锁	Update	IX 锁
X 锁	EXclusive	IX 锁
W 锁	Weak eXclusive	IX 锁
NS 锁	Next key Share	IS 锁
NW 锁	Next key Weak exclusive	IX 锁

表 19.1 中的第 3 列，是在获得此行级锁之前需要预先获得的表级锁，这里列出的是最低要求。

6 个行级锁的主要功能如下。

（1）S 锁：锁拥有者和任何其他并发的事务都可以读（但不能更改）锁定行中的数据。只要表不是使用共享锁锁定的，那么该表中的单个行可以使用共享锁锁定。但是，如果表是用共享锁锁定的，则锁拥有者不能在该表中获取行级别的共享锁。

（2）U 锁：此行级锁的拥有者可以读取该行的信息，如果要更新该行，则仍然需要一个行级的 X 锁；其他的应用只能读取该行的信息；此锁主要是用于 FOR UPDATE 的游标。

（3）X 锁：此行级锁的拥有者可以更新该行的记录，其他的应用不能连接此行的信息。

（4）W 锁（弱互斥，Weak Exclusive）：此锁与 X 锁类似，不同之处是此锁和 NW 锁兼容。锁的拥有者可以读取和更改被锁定的行。当向非目录表中插入行时，该行上将获得这种锁。

（5）NS 锁（下一键共享，Next Key Share）：类似于 S 锁，用于 New Key。锁拥有者和所有并发的事务都可以读（但不能更改）锁定行中的数据。这种锁用来在使用“读稳定性”或“游标稳定性”事务隔离级别读取的数据上代替共享锁。

（6）NW 锁（下一键互斥，Next Key Weak Exclusive）：类似于 W 锁，用于 Next Key。锁的拥有者可以读取但不能更新被锁定的行。当向非目录表的索引插入行时，表中下一行上就获得这种锁。

在 DB2 数据库中，是通过行级锁和表级锁协调作用来提供较好的并发性的，同时保证行或表中数据的安全。在 DB2 中默认情况下使用行级锁（当然需要 IS/IX 锁配合），只有当出现锁资源不足，或者是用命令指定使用表级锁的情况下，才会在应用连接中使用表级锁。

19.4.3 获取不同级别的锁的方法

在大多数情况下，DB2 数据库管理器在需要锁时，会隐式获取对应级别的锁，并掌握着这些锁的控制权。因此，一般来说，用户没有必要在应用程序中针对某一数据库对象显式定义锁。实际上，在 DB2 中，用户只能针对“表”显式地定义锁。例如：

```
LOCK TABLE table_name IN <SHARE | EXCLUSIVE> MODE
```

在默认情况下，DB2 数据库管理器总是尝试在事务中用行锁来保证一致性，以尽量减少被锁的资源。但用户可以更改这种默认的行为，使用的语句是：

```
ALTER TABLE table_name LOCKSIZE <ROW | TABLE>
```

19.5 隔离级别

事务指定一个隔离级别（ISOLATION LEVEL），该隔离级别定义一个事务必须与其他事务进行的资源或数据更改相隔离的程度。隔离级别从允许的并发副作用（例如，“脏读”或不可重复读）的角度进行描述。

19.5.1 隔离级别的种类

隔离级别分为RR/RS/CS/UR这4个级别，下面分别进行介绍。

1. 可重复读（Repeatable Read，RR）隔离级别

在此隔离级别下，DB2会锁住所有相关的记录。在一个SQL语句执行期间，所有执行此语句扫描过的记录都会被加上相应的锁。具体的锁的类型还是由操作的类型来决定，如果是读取，则加共享锁；如果是更新，则加独占锁。由于会锁定所有为获得SQL语句的结果而扫描的记录，所以锁的数量可能会很庞大，这时，索引的增加可能会对SQL语句的执行有很大的影响，因为索引会影响SQL语句扫描的记录数量。

2. 读稳定性（Read Stability，RS）隔离级别

此隔离级别的要求比RR隔离级别稍弱，此隔离级别下会锁定所有符合条件的记录。不论是读取，还是更新，如果SQL语句中包含查询条件，则会对所有符合条件的记录加相应的锁。如果没有条件语句，也就是对表中的所有记录进行处理，则会对所有的记录加锁。

3. 游标稳定性（Cursor Stability，CS）隔离级别

此隔离级别只锁住当前处理的记录。

4. 未提交读（Uncommitted Read，UR）隔离级别

在此隔离级别下，如果是读取操作，不会出现任何的行级锁。对于非只读的操作，它的锁处理和CS相同。

在这4种隔离级别中，CS是默认值。这4种隔离级别均可以保证DB2数据库在并发的环境下不会有数据丢失的情况发生。要注意的是，如果对记录进行了修改，需要在相应的记录上加独占类型的锁，这些独占类型的锁直到事务结束时才会被释放，这一点在4种隔离级别下都是相同的。

选择适当的隔离级别很重要：较低的隔离级别可以增强许多用户同时访问数据的能力，开销较少，但也增加了用户可能遇到的并发副作用（例如脏读或丢失更新）的数量。相反，较高的隔离级别减少了用户可能遇到的并发副作用的类型，但需要更多的系统资源，并要通过执行某种级别的锁定来完成此操作，而锁定可能会影响多用户系统中的其他用户。应平衡应用程序的数据完整性要求与每个隔离级别的开销，在此基础上选择相应的隔离级别。

19.5.2 设置数据库的事务隔离级别

数据库的隔离级别（Isolation Level）虽然控制着事务发生时数据库资源如何被锁定，但属性一般是在应用程序级别进行设置的。

设置数据库的事务隔离级别方法为：

（1）嵌入式的 SQL 程序，将会用到的隔离级别是在预编译（PREP）或绑定（BIND）时，通过设置 ISOLATION 选项来定义。

（2）对 CLI 接口的应用程序，隔离级别是在程序运行时通过调用 SQLSetConnectAttr()函数，设置 SQL_ATTR_TXN_ISOLATION 连接属性来定义；用户也可以通过修改 db2cli.ini 中的参数 TXNISOLATION 来定义。

（3）如果用户是通过 JDBC 和 SQLJ 来访问 DB2，则隔离级别的设置是在程序运行时通过调用 java.sql 连接接口的 setTransactionIsolation()函数来实现的。

（4）如果用户应用程序中没有设置隔离级别，则默认的隔离级别是游标稳定性（Cursor Stability）。

（5）如果用户用命令设置当前隔离级别，可以使用 CHANGE ISOLATION 命令。

CHANGE ISOLATION 命令的语法格式为：

```
CHANGE<SQLISL | ISOLATION> TO <CS|RR|RS|UR>
```

【例 19.3】 改变 MRKJ 数据库的隔离级别（**实例位置：光盘\TM\sl\19\2**）。

实现步骤如下。

（1）改变隔离级别前，必须终止后端进程：

```
TERMINATE;
```

（2）将隔离级别变成 UR，当连接至不支持 UR 的数据库时，会发生自动升级：

```
CHANGE ISOLATION TO UR;
```

（3）连接到目标数据库，此时该数据库的隔离级别已经改变：

```
CONNECT TO mrkj USER mr USING 1;
```

19.5.3 重写隔离级别（WITH 从句）

DB2 UDB 允许应用程序中的 SQL 语句指定锁绑定计划或包的隔离级别。这将有效地重写用于项目或包的隔离级别，但仅是在它出现的语句中。

用于重写隔离级别的 WITH 子句的格式为：

```
WITH<RR | RS | CS | UR>
```

WITH 子句可以用于以下类型的 SQL DML 语句：

☑ SELECT 语句。
☑ 搜索型 UPDATE 和搜索型 DELETE 语句。
☑ 子查询中的 INSERT 语句。

【例 19.4】 用 UR 隔离级别查询“成绩表”中“学号”为“14007”的学生信息（**实例位置：光盘\TM\sl\19\3**）。

实现步骤如下。

（1）改变隔离级别前，必须终止后端进程：

```
TERMINATE;
```

（2）连接到目标数据库：

```
CONNECT TO mrkj USER mr USING 1;
```

（3）在 SELECT 语句中采用 UR 隔离级别，不影响其他语句的隔离级别：

```
SELECT * FROM mr.成绩表 WHERE 学号='14007' WITH UR;
```

19.5.4 继承隔离级别

用于继承隔离级别的子句的格式为：

```
INHERIT ISOLATION LEVEL WITHOUT LOCK REQUEST
```

或

```
INHERIT ISOLATION LEVEL WITH LOCK REQUEST
```

19.5.5 锁避免和隔离

“锁避免”是在 DB2 V3 中引入的一种并发性技术。它使 DB2 UDB 在确定某页中的数据已经提交之后，就能够读取该页面，而无须获得锁。是否可以使用锁避免取决于应用程序在其 SQL 调用中使用的隔离程度。

RR 隔离级别和 RS 隔离级别不允许任何锁避免。带有 CS 隔离级别的游标对非限定的页面或行使用锁避免。带有 CS 隔离级别的歧义游标和只读游标对限定的和非限定的页面或行均使用锁避免。

19.6 小 结

本章主要讲解了数据库事务、事务日志记录、并发性控制、锁和隔离级别。在 DB2 中的锁曾经困扰了无数 DBA 和开发人员，相信每个初学者都有着不堪回首的痛苦学习经历。特别是在早期版本中，

由于 DB2 提供的工具和方法有限，使得锁问题的诊断和分析变得非常困难。随着 DB2 版本的演变，提供的锁监控和分析工具越来越多，在性能方面也有一些改进，特别是 V9.7 版本引入的当前已落实特性可以大大提高并发性能。

19.7 实践与练习

（1）简述事务的 4 大特性。（**答案位置：光盘\TM\sl\19\4**）

（2）DB2 确保数据一致性和可恢复性所使用的主要机制是什么？（**答案位置：光盘\TM\sl\19\5**）

（3）分别说明什么是锁、锁模式、锁粒度、锁升级？（**答案位置：光盘\TM\sl\19\6**）

（4）简述暂挂、超时、死锁的基本概念。（**答案位置：光盘\TM\sl\19\7**）

丛书历史与荣誉



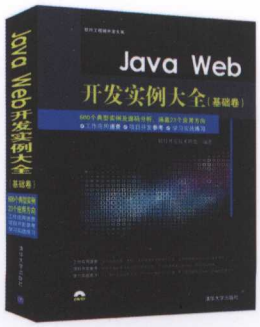
“软件开发视频大讲堂”丛书自2008年出版，已经过两次改版，因其编写细腻、易学实用、配套资源完善（包括全程实例视频讲解、实例源程序、教学课件PPT等）而备受读者欢迎，多个品种被近百所高校计算机相关专业、软件学院选为教学参考书，并连续八年在全国软件开发类零售图书排行榜中名列前茅。

推荐阅读



实例类图书

《Java Web开发实例大全》的基础卷、提高卷分别精选了约600个实例和约600个秘笈心法，涵盖了Java Web编程多个方面的各种应用，是目前市场上最全面的软件开发实例类丛书（丛书名：软件工程师开发大系）中的品种，该丛书堪称软件开发实例的“四库全书”，开发中所需的技术、技巧在本丛书中几乎都可以找到。



软件开发视频大讲堂

RUANJIANKAIFASHIPINDAJIANGTANG



清华大学出版社数字出版网站

WQBook 书文局泉

www.wqbook.com

ISBN 978-7-302-45800-5

定价：79.80元

（附1DVD，含视频讲解、实例资源库